Spectrum PLUS

Manual del usuario



PROLOGO

Muchas personas tienen de los ordenadores la idea de que son unas máquinas misteriosas, casi mágicas, cuyo complicadísimo funcionamiento solo puede ser comprendido por mentes privilegiadas; en cuanto a la programación de los mismos, pueden llegar a considerarla más como brujería que como un ejercicio intelectual.

Por otro lado, hay bastante gente que opina que los ordenadores personales no son realmente útiles y que son más unos juguetes caros que otra cosa.

Por supuesto, tanto unos como otros estan equivocados; el funcionamiento de un ordenador es algo facilmente comprensible, ya que , al fin de al cabo, es un funcionamiento LOGICO. En un ordenador se realizan, simplemente, una serie de operaciones en un orden determinado, con la ventaja de que esto se efectua muchísimo mas rapido. El programador no hace mas que indicarle a la computadora las operaciones que ha de realizar y en que orden deben realizarse. Por supuesto, poder hacer esto implica un periodo de aprendizaje; al ordenador hay que hablarle en su lengua y aprenderla no es mas que una cuestion de paciencia e interés.

Este manual ha sido confeccionado pensando en aquellos que se inician en el mundo de la informática personal; si Vd. es uno de ellos, esperamos que su lectura le ayude a entender mejor el ordenador que ha adquirido. El libro le puede parecer un poco "ladrillo", pero hemos preferido confeccionar una auténtica ayuda para el usuario antes que un bonito folleto a todo color.

Si por el contrario, ya esta iniciado en la programación, puede utilizarlo como libro de consulta y !!quien sabe!! quiza encuentre en sus paginas algun dato que desconocía.

Al final del libro se encuentra una lista con el juego de caracteres que utiliza el Spectrum, que le sera muy útil cuando realice sus propios programas. Asimismo, pruebe a ejecutar los programas ejemplo incluidos en los diferentes capitulos, y lo que es mas importante, atrévase a modificarlos. No tenga miedo de experimentar con su Spectrum; no lo estropeará. Cuantos más programas propios realice, más avanzará en el conocimiento de la programación.

Agosto 85

La biblioteca de programas comerciales para el ZX Spectrum es la mas extensa del mercado: existen mas de 10.000 títulos que abarcan desde los juegos de todo tipo a completos programas de gestión, cálculo, científicos, educativos, etc. Como ejemplo, diremos que este libro ha sido escrito utilizando un Spectrum con un programa de proceso de textos.

A medida que vaya adquiriendo conocimientos, podra comprobar que las posibilidades de su Spectrum estan mas allá del simple juego; esperamos que su nuevo ordenador le de las satisfacciones que espera y recuerde que solo existe un limite para lo que Vd. puede realizar: su imaginación.

INTRODUCCION

COMO CONECTAR SU SPECTRUM

Efectúe las siguientes operaciones:

- Conecte la fuente de alimentación (que se suministra con el equipo) al Spectrum.

- Conecte el cable de red de la fuente de alimentación a la red de 220 voltios.

- Conecte el Spectrum al televisor mediante el cable adjunto, un extremo al ordenador y el otro a la conexión de antena del televisor.

- Encienda el televisor y sintonicelo aproximadamente en el canal 36 de UHF hasta que aparezca claramente en pantalla el mensaje "1982 Sinclair Research Ltd.".

- Baje el volumen del televisor al mínimo.

- Si el televisor no reproduce bién los colores, retoque la sintonía del mismo.

- Para conectar el cassette, conecte el enchufe EAR del Spectrum con la salida de auriculares o altavoz externo del cassette, y la toma MIC del Spectrum a la entrada de micrófono de la grabadora, utilizando los cables que vienen incluidos con el equipo.

No todos los cassettes sirven para funcionar con el ordenador; lea el capítulo correspondiente del libro para saber cuales son los idoneos.

CAPITULO 1

EL TECLADO DEL SPECTRUM+

El teclado del Spectrum+ posee 17 teclas mas que el del Spectrum normal, ya que muchas de las funciones que necesitan de la pulsación de dos teclas simultaneamente poseen tecla propia en e Spectrum+. Asi pués se dispone de 57 teclas (aparte la barra espaciadora) que permiten el manejo de más de 200 letras, signos y comandos. Esto puede hacer pensar que su manejo es muy complicado, pero no es así.

Cada tecla tiene cinco funciones distintas; dentro de la misma hay escrita una palabra clave BASIC y debajo de esa palabra puede escontrarse o bien otra palabra o un símbolo. En la parte mas inferior de la tecla se encuentra la letra correspondiente a la misma.

Encima de la tecla (fuera de la parte sobresaliente de la misma) se hallan escritas dos palabras clave (o una palabra clave y un símbolo). A todas estas funciones se puede acceder, bién pulsando las teclas de ayuda CAPS SHIFT y SYMBOL SHIFT (que se hallan duplicadas a ambos lados del teclado) o bién cambiando el "modo" de operación en que se encuentra la máquina. Estos modos són:

El modo "K": Este modo se activa automaticamente al principio de una linea de programa y no se desactiva hasta que no se haya introducido una palabra clave. También se conecta despues de una sentencia THEN o de un signo ":". En este modo, al pulsar una tecla se obtiene la palabra clave BASIC escrita en blanco en la parte superior de la tecla (dentro del saliente). Inmediatamente despues de la impresión del comando correspondiente se activa automaticamente el modo "L".

El modo "L": Al pulsar una tecla, se obtiene la letra minúscula correspondiente. Si pulsamos CAPS SHIFT al mismo tiempo, obtendremos la misma letra, pero mayúscula. Por otro lado, si presionamos SYMBOL SHIFT y la tecla simultaneamente, obtendremos el símbolo o palabra escrito en la parte central de la tecla.

El modo "C": Se activa pulsando la tecla CAPS LOCK situada en la parte inferior izquierda del teclado. En este modo, todas las letras que se impriman seran mayusculas. Los signos y palabras clave escritas en el centro de las teclas se obtienen de la misma manera que en el modo "L".

El modo "G": Se obtiene pulsando la tecla GRAPH que se encuentra en la parte superior izquierda del teclado. Al activar este modo, podran obtenerse los símbolos gráficos impresos en las teclas numéricas y aquellos caracteres gráficos que hayamos creado nosotros mismos, situados en las teclas de la A a la U.

El modo "E": Se activa pulsando la tecla EXTEND MODE situada a la izquierda del teclado. De esta manera tendrá acceso a las palabras y signos situados fuera del saliente de las teclas. Para obtener la palabra escrita encima, simplemente pulse la tecla correspondiente. Si desea imprimir la palabra de la parte inferior, debe pulsar SYMBOL SHIFT al mismo tiempo. Una vez que la palabra o signo ha sido impreso, se vuelve automaticamente al modo "L".

Para pulsar dos teclas al mismo tiempo, lo mejor es utilizar el siguiente método: Se pulsa la tecla SHIFT que corresponda y MANTENIENDOLA PULSADA se aprieta la tecla deseada. Esto le evitará muchos fallos al teclear. Tenga en cuenta que si mantiene pulsada una tecla durante más de dos segundos, empezará la autorrepetición del caracter correspondiente.

INTRODUCCION Y EDICION DE LINEAS

Todo lo que vaya tecleando aparecerá en la parte inferior de la pantalla. Por delante de lo qúe vá escribiendo está situado un cursor en forma de cuadrado centelleante que le indica el modo en que se encuentra la maquina. Si pulsa DELETE (situada en la prte superior izquierda del teclado) el cursor retrocederá, borrando la ultima letra o comando que se encuentre inmediatamente detrás del mismo. Usando las teclas de cursor (marcadas con flechas en la parte inferior del teclado) podra desplazar el cursor a lo largo de la linea sin borrar nada. Para borrar la linea completa pulse EDIT (está situada debajo de GRAPH).

Si lo que esta tecleando es una linea de programa (lleva un número al principio), al pulsar **ENTER** la linea pasará a la parte superior de la pantalla. Al mismo tiempo, entre el numero de linea y la primera sentencia aparecerá el signo ">". A este signo se le denomina **"cursor de programa"** y la linea que lo lleva se denomina "linea en curso". A medida que vaya tecleando e introduciendo lineas, estas pasarán a la parte superior de la pantalla, situandose en el orden determinado por sus números de linea.

El cursor de programa estará situado normalmente en la última linea introducida. Puede colocar este cursor en la linea que quiera utilizando las teclas de cursor Pulsando **EDIT** la linea en curso aparecerá en la parte inferior de la pantalla y podrá ser modificada de la manera anteriormente mencionada.

Aparte de las teclas mencionadas hasta ahora, existe tecla independiente para los signos ; " , .

PRESENTACION VISUAL

La pantalla del televisor tiene 24 lineas y 32 columnas ; las 22 primeras lineas muestran los listados de los programas o los efectos de la ejecucion de los mismos.

Cuando la impresión supera las 22 lineas, todo el texto se desplaza una linea hacia arriba; antes de que desaparezca la primera linea por la parte superior, aparece en la parte baja el mensaje **"scroll?". Si se pulsa n, SPACE o STOP el texto no se desplaza** hacia arriba y aparece un mensaje de error "D BREAK - CONT repeats". Cualquier otra tecla hace que continue el desplazamiento vertical.

Las dos ultimas lineas se utilizan para introducción de lineas de programa y comandos directos, asi como para mostrar los mensajes de error. Ocasionalmente se puede escribir en ellas.

CAPITULO 2

EL BASIC

El auténtico cerebro del ordenador es el microprocesador o CPU (abreviatura de Central Processing Unit - Unidad Central de Proceso). La CPU se encarga de recibir y procesar los datos y de ejecutar las operaciones correspondientes en función de los mismos.

Los datos llegan a la CPU en forma de impulsos eléctricos de frecuencia y duracion variable. El microprocesador interpreta esos impulsos como datos. Por ejemplo, si en un momento dado recibe un impulso, lo interpreta como un 1, y si no recibe impulso lo interpreta como un 0. Estas agrupaciones de ceros y unos son interpretadas como números por el microprocesador, que de esta manera recibe los datos en forma numérica. Esto implica que toda la información debe ser codificada en forma de números, y esos números en forma de agrupaciones de unos y ceros. A este sistema de numeración se le denomina BINARIO.

Se llama "lenguaje máquina" a la manera que tiene el procesador de interpretar los impulsos que le llegan. Por supuesto que codificar todas las instrucciones de nuestros programas en forma de números binarios sería tediosa y dificil, por no decir imposible.

Es por ello que se inventaron los lenguajes de programación. En realidad éstos són programas en lenguaje máquina, que se encargan de la codificación de las instrucciones que le damos para que sean comprensibles por la máquina. Los lenguajes de programación llevan incorporado un conjunto de instrucciones de facil comprensión, que són las que utilizamos al realizar nuestros programas. A estos lenguajes se les llama "de alto nivel" por estar mas cerca de la comprensión del usuario que de la de la máquina.

El lenguaje de programación del Spectrum es el BASIC. Este es un lenguaje ampliamente difundido en la mayoría de ordenadores personales debido a su facil comprensión y gran versatilidad. Lamentablemente, cada ordenador incorpora su propia versión de BASIC; esto hace que programas escritos en BASIC en un ordenador, no puedan ejecutarse en otro distinto.

El BASIC está directamente derivado del inglés, es decir, muchos comandos de BASIC tienen el mismo significado que la palabra inglesa correspondiente.

EL BASIC DEL SPECTRUM

El BASIC que se emplea en el Spectrum es muy sencillo de entender y utilizar. Para introducir los comandos no hace falta deletrearlos, sino que basta una pulsación de la tecla correspondiente. A estos comandos se les denomina "TOKENS". Algunas de estas instrucciones se encuentran en todas las versiones de BASIC (PRINT, INPUT, RUN...) y otras són exclusivas (INK, INVERSE, OVER...). Tanto unos como otros serán descritos con detalle en este libro.

ESTRUCTURA DE LA PROGRAMACION

El Spectrum permite la introducción tanto de líneas de programa como de comandos directos. Por ejemplo, teclee lo siguiente:

```
PRINT "pepito"
```

y pulse la tecla ENTER. La palabra "pepito" aparecerá en la parte superior de la pantalla y en la parte inferior de la misma se imprimirá el mensaje "O OK, O:1" indicativo de que se ha ejecutado la instrucción correctamente. si hubiera tecleado

```
PRINT ppepito"
```

no se hubiera ejecutado la instrucción; en su lugar habría aparecido un signo "?" parpadeante, indicativo de que se ha producido un error. Introduzca ahora

```
10 PRINT "pepito"
```

(recuerde que ha de pulsar ENTER). La linea no se ejecutará, sino que aparecera en la parte superior de la pantalla, con el signo ">" (cursor de programa) entre el numero de linea y la palabra PRINT. El ordenador ha almacenado la linea en su memoria para ejecutarla mas adelante. Si desea que se ejecute ahora, teclee

```
RUN
```

Ahora introduzca

```
5 PRINT "hola"
```

a pesar de que esta linea ha sido introducida despues que la anterior, aparecerá en la pantalla ANTES. La causa de ello es que su numero de linea es inferior. Recuerde que las lineas se listarán en la pantalla en el orden indicado por su numero de linea.

Habra comprobado que siempre escribimos entre comillas las palabras que deseamos imprimir. Esto no solo debemos hacerlo con palabras sueltas, sino con cualquier grupo de ellas que deseemos imprimir. A este conjunto de letras que vá entre comillas se le denomina "cadena de caracteres" o, simplemente "cadena".

Podemos prescindir de las comillas cuando lo que queremos imprimir sean números. Ejemplo:

```
15 PRINT 24
```

Al ejecutar este programa (pulse RUN), obtendremos:

```
hola
pepito
24
```

y en la parte inferior de la pantalla el mensaje "0 OK, 15:1". Observe que este mensaje es diferente del que aparecía cuando ejecutábamos un comando directo. Los números que aparecen detras de "OK," corresponden al número de linea y de sentencia en donde se ha detenido el programa, respectivamente.

El numero 24 que hemos escrito en la linea 15 es evaluado por el ordenador como un valor numérico; si lo hubiesemos colocado entre comillas habría sido tratado como una cadena de caracteres. Esto es muy importante, ya que el ordenador solo puede realizar operaciones matemáticas con valores numericos y no con cadenas.

Veamos un ejemplo. Teclee el siguiente comando directo:

```
PRINT 5+5
```

el resultado sera 10. Este numero se imprimirá en la parte superior de la pantalla, Ahora introduzcamos:

```
PRINT "5"+"5"
```

como ve, el valor cambia completamente. Aunque las cadenas pueden sumarse, lo hacen de una manera muy especial. Ejecute:

```
PRINT "Spec"+"trum"
```

para ilustrar este principio.

Vamos ahora a cambiar la linea 15. Podríamos volver a introducirla completamente, pero existe un método mejor. Como podemos ver, el cursor de programa esta situado en dicha linea (si no lo estuviera, lo situaríamos por medio de las teclas de cursor). Pulsando **EDIT** la linea se duplicará en la parte inferior de la pantalla. Situemos el cursor al final de la linea (teclas de cursor) y tecleemos

```
: PRINT "fin del programa"
```

al pulsar ENTER, veremos como la linea modificada está situada en el lugar correcto en el programa. Al ejecutarlo, veremos que el mensaje es ahora "0 OK, 15:2" indicativo de que el programa se ha detenido en la linea 15, sentencia 2.

Para borrar una linea de programa, basta teclear su numero (seguido de ENTER) y la linea desaparecera. Si escribimos una linea con un numero igual al de una linea ya existente, dicha linea sera sustituida por la nueva. Al borrar alguna linea, el cursor de programa "desaparecerá". Pulse la tecla de cursor que tiene su flecha hacia arriba y el cursor aparecerá en la linea anterior a la que ha sido borrada.

Hasta ahora, el listado del programa ha aparecido automaticamente al pulsar ENTER despues de la ejecución del mismo. Existe un comando cuya unica función es producir listados, que es LIST. Teclee:

```
LIST 10
```

En la pantalla apareceran las lineas 10 y 15, con el cursor de programa situado en la primera de ellas. Aunque la linea 5 no aparezca en pantalla, continúa estando en el programa.

La utilidad del comando LIST es evidente cuando se trata de programas largos, en los que se quiere colocar el cursor de programa en una linea determinada, para proceder a su edición. En el caso de que el listado de un programa no quepa en la pantalla, aparecera el conocido mensaje "scroll?".

Una practica muy saludable cuando de escriben programas es introducir comentarios aclaratorios en su listado. Esto se hace utilizando el comando REM. Cuando el ordenador encuentra una linea con una sentencia REM, ignora dicha linea y ejecuta la siguiente. Es por ello que despues de una sentencia REM no se debe introducir ninguna instrucción, ya que el ordenador la ignorará.

Una manera de borrar un programa es desconectar el ordenador. Pero lo normal es que se utilice el comando NEW. Esta instrucción borra el programa BASIC y las variables que hayamos podido definir. Tecléelo ahora.

Introduzca el siguiente programa:

```
10 REM impresion continua
20 PRINT "conozca su Spectrum"
30 GO TO 20
```

El comando GO TO contenido en la linea 30 sirve para indicar al ordenador que debe saltar a la linea especificada y continuar alli la ejecución del programa.

Pulse RUN para que corra el programa; la frase "conozca su Spectrum" se imprimirá 22 veces y luego aparecerá un mensaje "scroll?". Pulse cualquier tecla (menos N o SPACE) y la frase se imprimirá otras 22 veces (no lo notará, ya que aunque las frases se desplazan hacia la parte superior de la pantalla, son todas iguales) y volvera a preguntar "scroll?". Si entonces pulsa N o SPACE se imprimirá un mensaje "D BREAK - CONT repeats 20:1". También puede detener el programa pulsando BREAK (CAPS SHIFT + SPACE) antes de que aparezca el mensaje "scroll?". En ese caso el mensaje que aparecera sera "L BREAK into program, 20:1".

Instroduzca ahora la siguiente linea

```
25 PRINT "facilmente"
```

y ejecute el programa. Cuando este se pare con el mensaje "scroll?" pulse N o SPACE para obtener el mensaje "D BREAK - CONT repeats". Observe que las dos ultimas lineas mostradas en pantalla se leen

```
conozca su Spectrum
facilmente
```

pulse ahora el comando CONTINUE (esta situado en la tecla C y se encuentra abreviado como CONT). El programa continuará ejecutandose, ya que el efecto de esta sentencia es precisamente ese. Al pulsar CONTINUE se repite la instrucción en donde se detuvo el programa y este continúa ejecutandose a partir de ella.

Hay un caso excepcional: si se detiene el programa pulsando BREAK, al pulsar CONTINUE continuará la ejecución en la sentencia que siga a aquella donde se detuvo el programa. Esto es especialmente útil en el caso de que el programa contenga errores; si el programa se detiene a causa de un error, podrá subsanarse este y seguir con la ejecucion por medio de CONTINUE.

Otro comando muy util es STOP. Esta instrucción detiene permanentemente el programa, apareciendo un mensaje "9 STOP statement". Pulsando CONTINUE, el programa seguira ejecutandose a partir de la linea siguiente a aquella que contiene el STOP. Para comprobar esto, escriba

```
10 PRINT "esta es la primera parte"
20 STOP
30 PRINT "esta es la segunda parte"
```

y ejecutelo. Cuando aparezca el mensaje, pulse CONTINUE.

En el proximo capitulo hablaremos sobre mas usos de la sentencia STOP.

Los programas BASIC que hemos visto hasta ahora se ejecutan sin interrupción. Sin embargo, es posible deneter la ejecución momentaneamente por medio del comando PAUSE. Esta sentencia se usa junto con un número que expresa el tiempo de pausa en cincuentavos de segundo. Así, PAUSE 50 detiene la ejecución durante un segundo. Para detener el programa hasta que se pulse una tecla use PAUSE 0.

Ya hemos visto, pues, las sentencias PRINT, RUN, LIST, NEW, GO TO, CONTINUE, REM, PAUSE, STOP y BREAK. Las 6 primeras pueden usarse tanto como comandos directos como dentro de un programa, mientras que REM, PAUSE y STOP solo pueden usarse dentro de un programa y BREAK solo como comando directo.

CAPITULO 3

LAS VARIABLES

En el capitulo anterior veíamos como imprimir en pantalla palabras o números. Hasta ahora solo hemos manejado valores constantes, es decir, definidos a priori y no modificables. Ahora vamos a tratar otro tipo de valores que pueden ser modificados a lo largo de la ejecución del programa; son las variables.

Para entender el concepto de variable, supongamos que tenemos un gran mueble lleno de cajones; éstos pueden tener una etiqueta exterior que los identifique, de manera que seamos capaces de localizarlos rapidamente.

Ahora, metamos algo en un cajón; por ejemplo, diez garbanzos, y etiquetemos el cajón con la letra "a". Introduzcamos en otro cajón tres garbanzos e identifiquemoslo como "b". De esta manera, cuando vayamos a uno u otro cajón, encontraremos el número de garbanzos que hayamos introducido previamente.

Entonces, si en el cajón "a" introducimos tres garbanzos mas, ya no tendremos 10 garbanzos en ese cajón, sino 13. Podemos decir que el contenido de ese cajón ha VARIADO. Lo mismo podemos hacer con el contenido de los otros cajones.

Pues bien, en el ordenador tambien podemos "etiquetar" ciertas partes de la memoria e introducir valores que pueden ser cambiados a voluntad. La máquina almacena estos valores e introduce las modificaciones que nosotros le indiquemos. Estas zonas de la memoria que definimos se llaman variables.

Existen dos tipos de variables: numéricas y alfanuméricas. A éstas ultimas tambien se les llama "de cadena" ya que no almacenan un número sino una cadena de caracteres.

DEFINICION DE VARIABLES

Para definir una variable, tanto numérica como de cadena, se emplea el comando LET. Ejemplo:

```
LET a=25
```

si ahora escribimos

```
PRINT a
```

aparecerá impreso el numero 25. Pruebe ahora

```
LET a=a+1
```

y luego

```
PRINT a
```

Como puede ver, el valor de la variable "a" ha cambiado a 26.

Es necesario definir una variable (etiquetarla) para que pueda ser utilizada. Si teclea

```
PRINT b
```

aparecerá un mensaje de error "2 Variable not found, 0:1" (no se encuentra la variable).

La variable "a" es numérica; le hemos dado un nombre de una sola letra, pero no es necesario que sea así siempre. De hecho, el nombre puede constar de hasta 128 letras o digitos (también se admiten espacios), siempre que el primer caracter no sea un dígito. Ejemplos:

coco
este libro le aburrira
a1
A1

En los dos últimos casos, los dos nombres se refieren a la misma variable.

Ahora veamos ejemplos de nombres no permitidos:

1200 (empieza con un digito)
ocho-cuatro ("-" no es una letra ni un digito)

Al contrario de las numericas, las variables de cadena solo pueden tener un nombre compuesto por una sola letra y el signo "$". La cadena a introducir debe estar entre comillas,

```
LET a$="PEPE"
```

Sólo se puede operar matematicamente con las variables numéricas. Las alfanuméricas pueden sumarse, pero el resultado de esa suma no es matemático, como ya se vio en el capitulo anterior.

Una cadena puede contener cualquier caracter, pero puede existir una pega en el caso de que se quieran incluir comillas en la misma. Pruebe:

```
PRINT "Hoy he visto "TRON" y me ha gustado"
```

Aparecera el signo de interrogación parpadeante, indicativo de que se ha cometido un error. Esto es porque el ordenador cree que la cadena ha terminado despues de la palabra "visto" y no sabe interpretar lo que viene después. Para evitar esto, escriba:

```
PRINT "Hoy he visto ""TRON"" y me ha gustado"
```

No se preocupe; las comillas solo se imprimirán una vez. Esto es válido tanto para una cadena como para una variable alfanumérica.

Hasta ahora solo hemos visto como se definen variables desde dentro de un programa; pero si queremos definirlas durante la ejecución del mismo (por ejemplo, para introducir datos con los que ha de operar el programa) necesitamos un nuevo comando: la instrucción INPUT.

Una sentencia INPUT tiene la siguiente forma:

```
INPUT nombre
```

para definir una variable alfanumérica, o

```
INPUT letra$
```

para una variable de cadena. Cuando el ordenador encuentra una sentencia INPUT detiene el programa y un cursor parpadeante aparece. Teclee entonces el valor (o cadena) que desee y pulse ENTER. La variable quedara definida y el programa continuara su ejecucion.

En el caso de que la variable a definir sea alfanumérica, el cursor aparecera entre comillas.

Veamos ahora un ejemplo que nos ilustre lo que hemos visto hasta ahora. Borre el ordenador (mediante NEW) e introduzca el siguiente programa:

```
10 REM programa sumador
20 LET suma=0
30 LET a$="dime un numero"
40 PRINT a$
50 INPUT numero
60 LET suma=suma + numero
70 PRINT suma
80 GO TO 40
```

Ejecute el programa con RUN (o con GO TO 10) y vea lo que sucede.

Como puede ver, no es necesario volver a la linea 30 por medio de GO TO, ya que la variable a$ permanecerá definida hasta que apaguemos el ordenador. 'Que pasaría si cambiamos la linea 80 por GO TO 20?.

Para detener el programa, cuando aparezca el cursor teclee STOP en vez de un número. El programa se detendrá con el mensaje "H STOP in INPUT, 50:1".

Una advertencia: las variables se borrarán si emplea el comando RUN y tendrá que volver a definirlas. Si desea evitar ésto, use GO TO en vez de RUN para ejecutar los programas.

CAPITULO 4

LOS BUCLES FOR-NEXT

Supongamos que queremos que el ordenador realice una tarea repetidas veces. Por ejemplo, que imprima una cadena o nos pida un dato varias veces. Una forma es hacerlo "a mano":

```
10 LET a=0
20 INPUT b
30 LET a=a+b
40 GO TO 20
```

Esto es válido cuando la cantidad de números a introducir es ilimitada. Pero la cosa cambia cuando queremos que la instrucción se ejecute solo un cierto número de veces. La mejor manera de hacer ésto es por medio del conjunto de comandos FOR-NEXT. De esta forma, si quisiéramos sumar cinco numeros, el programa se escribiría:

```
10 LET a=0
20 FOR n=1 TO 5
30 INPUT b
40 LET a=a+b
50 NEXT n
60 PRINT a
```

El número de veces que se repite la acción se define en la llamada "variable de control" del bucle (en este caso "n"). El nombre de esta variable debe estar constituído obligatoriamente por una sola letra (mayúscula o minúscula). No debe existir ninguna otra variable con el mismo nombre, aunque sí se puede emplear dicho nombre para todas las variables de control de todos los bucles del programa, siempre y cuando no haya dos bucles "imbricados" (dependiendo el uno del otro).

El ordenador incrementa el valor de la variable de control en una unidad cada vez que se ejecutan las instrucciones incluidas en el bucle. Cuando las mismas se han ejecutado el numero de veces prescrito, el programa continúa su ejecución en la sentencia que sigue al comando NEXT.

La variable de control puede utilizarse en las sentencias incluidas en el bucle.

Por ejemplo, este programa imprime en pantalla la tabla de multiplicar del número que se le introduzca:

```
10 FOR n=1 TO 10
20 INPUT a
30 PRINT n,a*n
40 NEXT n
```

Como hemos dicho antes, la variable de control incrementa su valor en una unidad cada vez que el bucle da una "vuelta". Pero esto no tiene que ser siempre así. El comando STEP permite especificar la cantidad en que debe cambiar el valor de dicha variable.

Pruebe este programa:

```
10 FOR n=1 TO 20 STEP 2
20 PRINT n
30 NEXT n
```

Siempre es necesario emplear STEP cuando el incremento del bucle sea negativo.

```
10 FOR n=20 TO 1 STEP -1
20 PRINT n
30 NEXT n
```

Los bucles FOR-NEXT no se pueden mezclar arbitrariamente, sino que deben estar uno completamente dentro del otro. Por ejemplo:

```
10 FOR i=1 TO 10
20 FOR j=1 TO 5
30 PRINT i*j;" ";
40 NEXT j
50 NEXT i
```

esta es la disposición correcta. La erronea sería:

```
10 FOR i=1 TO 10
20 FOR j=1 TO 5
30 PRINT i*j;" ";
40 NEXT i
50 NEXT j
```

Las sentencias de un bucle FOR-NEXT no tienen por que estar en lineas diferentes.

```
10 FOR i=1 TO 10: PRINT 2*i: NEXT i
```

En este caso solo se ejecutarán las sentencias situadas entre el FOR y el NEXT.

El error "1 NEXT whithout FOR" aparece si se ha saltado al interior de un bucle desde el programa (por medio de GO TO). Evitelo siempre.

CAPITULO 5

MATRICES

En el capítulo 3 veíamos como definir y utilizar variables. Siempre que deseemos almacenar o variar un dato previamente guardado, deberemos definir (o actualizar) las variables correspondientes.

Sin embargo, el metodo de asignación de variables que hemos visto puede ser francamente antipráctico y en la mayoría de las veces, muy pesado. Este sería el caso de que tuvieramos que introducir gran cantidad de datos; tendríamos que designar un nombre para cada uno, lo cual, aparte de tedioso, impediría que pudiéramos procesar dichos datos rápida y facilmente. El problema se agudiza en el caso de variables alfanuméricas, ya que, como hemos visto, solo podemos definir un máximo de 26 (el número de letras del alfabeto).

Asi pués , debemos utilizar otro método de asignación de datos; en lugar de guardarlos en variables, lo haremos en MATRICES.

Las matrices (o tablas) estan constituidas por un conjunto de variables, las cuales tienen el mismo nombre, pero se diferencian entre sí por un número (el subíndice).

Para dimensionar una matriz se emplea el comando DIM, de la siguiente forma:

```
DIM nombre (dimensiones)
```

Esta sentencia reserva parte de la memoria del ordenador para el almacenamiento de las variables de la matriz. Su uso es preceptivo.

Veamos un ejemplo: supongamos que queremos almacenar 100 numeros en el ordenador. Lo primero que debemos hacer es dimensionar la matriz que los contendrá. Hagamos:

```
10 DIM a (100)
```

si ejecutamos el programa y luego hacemos

```
PRINT a (1)
```

obtendremos el valor cero. Esto es porque al dimensionar una matriz numérica, todas sus variables toman dicho valor.

Si lo que queremos almacenar son cadenas en vez de números, tendremos que definir una matriz alfanumérica. Siguiendo con el ejemplo anterior, para almacenar 100 nombres, haríamos

```
20 DIM a$ (100,10)
```

como se ve, esta vez es necesario definir dos dimensiones para la matriz. El segundo número expresa la longitud máxima de la cadena que se vaya a introducir, en este caso 10 caracteres. Si la misma es mas larga que la dimension establecida, sera recortada en su extremo final, y si es demasiado corta, se añadirán espacios al final de la misma.

De ésto se deriva que la matriz siempre ocupará el mismo espacio en memoria, sea cual sea la longitud de la cadena que introduzcamos.

Una vez hemos dimensionado las matrices, hemos de dar los correspondientes valores a las variables que las componen. Una manera de hacerlo es por medio de LET o INPUT, pero esto podría resultar demasiado largo y tedioso. La manera mas rápida y eficiente de hacerlo en utilizando los comandos READ, DATA y RESTORE.

Una sentencia READ consta de un comando READ y una serie de nombres de variables, que deben estar separados por comas. A estas variables se les asignan los valores contenidos en una sentencia DATA (que puede estar situada antes o después que la sentencia READ). Los valores contenidos en la sentencia DATA también deben estar separados por comas y, si las variables son alfanuméricas, deben estar entre comillas.

Cada vez que se lee un dato, el ordenador prepara el siguiente dato de la lista para su lectura, colocando un "puntero" que señala hacia el dato a leer. Así pues, este puntero avanza automaticamente a través de la lista de datos hasta llegar al final de la misma.

Veamos un ejemplo explicativo. El programa que viene a continuación almacena 5 nombres y sus correspondientes números de teléfono

```
 10 DIM a$ (5,10): DIM a (5)
 20 FOR n=1 TO 5
 30 READ b$: LET a$(n)=b$
 40 NEXT n
 50 FOR n=1 TO 5
 60 READ b: LET a(n)=b
 70 NEXT n
100 DATA "Fernando","Pepe","Juan","Paco","Alberto"
110 DATA 2345454,2456789,3422134,5656789,2537824
120 DATA 4560214,3454567,2314536,2345434,4567890
```

Si se intentan leer mas datos que los contenidos en la sentencia DATA, se producirá un error. Aunque en la linea 120 figura otra sentencia DATA, solo seran leídos los numeros existentes en la linea 110, ya que esta es la primera sentencia DATA encontrada por READ y el puntero solo ha avanzado hasta el último dato de dicha linea.

Vamos a ampliar el programa introduciendo las siguientes lineas:

```
210 FOR n=1 TO 5
220 PRINT a$(n),a(n)
230 NEXT n
```

Ejecutemos el programa (RUN); aparecerán en la pantalla los nombres y sus correspondientes números. La coma de la linea 220 actúa como "separador"; indica al ordenador como debe imprimir los datos en la pantalla. Veremos esto con mas detalles en un próximo capitulo.

Se preguntará porque hemos escrito una segunda sentencia DATA. Sin borrar nada del programa, escriba

```
45 RESTORE 120
```

Ahora corra el programa; descubrira que los números han sido sustituidos por los contenidos en la linea 120. Esta es la función del comando RESTORE: indicar al comando READ a partir de que sentencia DATA debe empezar a leer; es decir, posiciona el puntero en un lugar determinado de la lista de datos.

Las matrices tambien pueden rellenarse mediante INPUTs (en el caso de programas de bases de datos), e incluso pueden grabarse y cargarse desde la cinta de cassette (o el microdrive).

CAPITULO 6

USOS ESPECIALES DE PRINT E INPUT
COMANDOS COMPLEMENTARIOS Y SEPARADORES

Vamos a ver en este capítulo las diversas formas en las que podemos complementar el comando PRINT para de esta manera definir formatos de impresión, tanto en pantalla como en impresora.

Lo primero que tenemos que saber es que el Spectrum define en la pantalla 24 lineas y 32 columnas. Como ya dijimos, las dos últimas lineas no son normalmente accesibles, ya que son las que emplea el ordenador para la ejecución de las sentencias INPUT y para mostrar los mensajes de error. Las posiciones de impresión se definen por medio de dos números; el primero de ellos corresponde a la linea y el segundo a la columna. Asi, 20,14 corresponde a la linea 20, columna 14.

Hasta ahora hemos utilizado PRINT sin especificar el lugar de la pantalla donde queríamos que actuara. Al encender el ordenador, o al haberse ejecutado una sentencia CLS, la posición de impresión es la 0,0. Cada vez que se ejecuta una sentencia PRINT (a menos que empleemos complementos en la misma) la posición de impresión se desplaza al principio de la siguiente linea. De esta manera podemos emplear el comando PRINT sin ningún tipo de argumento en el caso de que queramos imprimir una linea en blanco. Por ejemplo:

```
10 FOR n=1 TO 5
20 PRINT
30 NEXT n
40 PRINT "cinco lineas en blanco"
```

Existe un comando complementario que nos permite imprimir en la parte de la pantalla que deseemos: AT. Se utiliza así:

```
PRINT AT 11,3;"Fila 11, columna 3"
```

El primer caracter de la cadena se imprimirá en la posición de impresión especificada y los demas a continuación. El punto y coma que se emplea es un "separador"; efectivamente, algunos signos de puntuación permiten definir formatos de impresión. Esto funciona de la siguiente forma:

- ";" significa "imprimir inmediatamente despues"

- "," significa "desplazar posición de impresión 16 columnas"

- "'" significa "desplazar posición de impresión una linea"

Para comprobar esto, teclee el siguiente programa:

```
10 FOR n=1 TO 5
20 PRINT n,n*n
30 PRINT
40 NEXT n
```

Cambie el separador de la linea 20 y corra el programa.

Si se coloca un signo ";" al final de una sentencia PRINT, la posición de impresión no se desplazara a la siguiente linea. Pruebe:

```
10 FOR n=1 TO 10
20 PRINT n;
30 NEXT n
```

retire el punto y coma y vuelva a correr el programa para ver la diferencia.

La utilización de un separador repetidas veces multiplica su efecto. Borre el ordenador y teclee:

```
PRINT '''''"cinco lineas en blanco"
```

Tambien pueden combinarse varios separadores distintos

```
PRINT ''''',"cinco, dieciseis"
```

El comando TAB sirve para determinar la columna de impresión.

```
10 FOR n=1 TO 10
20 PRINT TAB 10;n
30 NEXT n
```

Como se ve, es bastante mas sencillo de utilizar que AT, aunque no es tan preciso. Su principal ventaja es a la hora de utilizar la impresora, ya que esta no reconoce AT pero si TAB. También es útil para la impresión de tablas.

Observe que tanto AT como TAB deben usarse con ";".

El comando AT y el separador ";" pueden utilizarse también con las sentencias INPUT, aunque con algunas limitaciones en su uso. Por ejemplo:

```
INPUT AT 1,0;n
```

imprime el cursor de INPUT en la segunda de las dos lineas de la parte inperior de la pantalla.

El separador tambien se utiliza en el caso de que se quiera introducir un texto junto con la sentencia INPUT.

```
INPUT "fecha? ";a$
```

OTROS USOS DE LA SENTENCIA INPUT

No solo se pueden introducir textos en las sentencias INPUT, sino tambien variables, aunque estas últimas deben ir entre paréntesis

```
10 LET a=25
20 INPUT("Mi numero es ";a);". El tuyo? ";b
```

Como vimos en el capitulo 3, cuando en una sentencia INPUT se define una variable alfanumérica, el cursor aparece entre comillas. Esto posibilita borrarlas para interrumpir el programa en ese punto por medio de STOP. Para evitarlo podemos utilizar el comando LINE de la siguiente forma:

```
INPUT LINE a$
```

el cursor aparecerá sin comillas, aunque la sentencia actuará como si estuvieran. Al no estar las comillas, no podrán ser borradas y, por lo tanto, detenido el programa.

En el caso de querer incluir textos en la sentencia INPUT LINE, el comando LINE irá despues del texto.

```
INPUT "Nombre? ";LINE a$
```

CAPITULO 7

FRAGMENTACION DE CADENAS

En todos los ordenadores existe la posibilidad de fragmentar las cadenas, es decir, de dividirlas y tratar cada trozo por separado. Por lo tanto, el Spectrum también puede hacerlo, aunque el metodo que utiliza difiere del que se emplea corrientemente.

Antes de nada, debemos aclarar que en los "trozos" o fragmentos de la cadena los caracteres que los compongan deben estar situados en el mismo orden que en la cadena de la que procedan. Por lo tanto, en la cadena "Spectrum" un fragmento podría ser "Spec" pero no "cum" ni "mur".

Para fragmentar cadenas se emplea el comando TO de la siguiente forma:

```
PRINT "cadena" (numero TO numero)
```

por supuesto no es necesario emplear siempre PRINT; puede usarse cualquier otro comando que trabaje con cadenas.

Si se omite el primer número, el ordenador le da el valor 1; si el que no se pone es el del final, su valor será el de la longitud de la cadena. Ejemplos:

```
"Spectrum" ( TO 4)="Spec"

"Spectrum" (3 TO )="ectrum"

"Spectrum" ( TO )="Spectrum"
```

Para tener una subcadena de un solo caracter, no se emplea TO, sino el número de orden de dicho caracter.

```
"Spectrum"(3)="e"
```

En caso de que uno de los números sea igual o menor que 0 o mayor que la longitud de la cadena se produce un error "3 Subscript Wrong" (subindice erroneo).

Tambien podemos utilizar el metodo de fragmentacion para cambiar partes de una cadena. Teclee:

```
LET a$="Prueba"
```

ahora

```
LET a$(3 TO 6)="ieto": PRINT a$
```

Una observacion: la cadena permanecerá con la misma longitud sea cual sea la longitud de la subcadena. Pruebe:

```
LET a$(3 TO 6)="obando": PRINT a$
```

en este caso la subcadena ha sido cortada por la derecha para ajustarse a la longitud de la cadena a$. En el caso de que hubiese sido más corta se habrían puesto espacios para completarla.

```
LET a$(3 TO 6)="oto"
```

en este caso a$ sería igual a "Proto ". A esta operación, que el ordenador realiza de forma automática, se le llama "asignación procusteana" (por el celebre posadero Procustes que, para que encajaran correctamente en las camas, estiraba a sus clientes en un potro de tormento o les cortaba los piés).

En capítulos posteriores veremos diversas funciones que tratan con cadenas.

CAPITULO 8

TOMA DE DECISIONES

Una de las características mas útiles de los ordenadores es su capacidad para decidir en función de que se cumplan o no unas condiciones. Las sentencias encargadas de esta función se llaman "sentencias condicionales". En el Spectrum (y en la mayoría de los ordenadores) las sentencias condicionales estan compuestas por los comandos IF y THEN, que asumen la siguiente forma:

```
IF condicion THEN sentencias
```

Si se cumple la condición se ejecutan las sentencias situadas despues de THEN. Inmediatamente despues de introducir THEN el cursor, que estaba en "L", cambia a "K", lo que permite la introducción de "tokens" (palabras clave). En el caso de que la condición no se cumpla dichas sentencias se ignoran y el programa continúa su ejecución en la siguiente linea. Esto último es muy importante; no deben colocarse detras de THEN mas que las sentencias que deban ejecutarse al cumplirse la condición.

Un ejemplo:

```
10 INPUT a
20 IF a=1 THEN PRINT "ha pulsado ""1""": STOP
30 PRINT "No ha pulsado ""1"""
```

La sentencia STOP de la linea 20 solo se ejecutara si la condición es verdadera. El signo "=" es una "relación" y es el que expresa la condición. Una relación puede comparar dos números o dos cadenas y pueden asociarse varias de ellas por medio de "operadores lógicos" (que estudiaremos en un próximo capítulo).

Las relaciones utilizables són:

> siginifica "mayor que"

< significa "menor que"

<= significa "menor o igual que"

>= significa "mayor o igual que"

= significa "igual a"

<> significa "distinto de"

Todos estos símbolos han de ser tecleados "de una vez", es decir, con una sola pulsación de tecla (cada uno tiene su tecla correspondiente).

Por supuesto que las sentencias IF-THEN no son siempre tan sencillas; veremos otras más complicadas en los capítulos que siguen.

CAPITULO 9

SUBRUTINAS

Cuando se escribe un programa puede darse el caso de que una instrucción (o conjunto de ellas) deba ejecutarse varias veces a lo largo del mismo.

Imaginemos que estamos imprimiendo textos y cuando se llene una pantalla queremos que el programa se pare y avise al usuario que pulse una tecla para ver otra pagina. Esto podria hacerse de la siguiente forma:

```
10 PRINT "una pagina de texto"
20 PRINT AT 21,1;"pulse una tecla para continuar"
30 PAUSE 0
40 CLS
50 PRINT "Otra pagina"
60 PRINT AT 21,1;"pulse una tecla para continuar"
70 PAUSE 0
80 CLS
90 PRINT "una pagina mas"
.
.
.
.
```

Como puede verse, este sistema, aunque correcto, sería bastante tedioso, aparte del gasto extra de memoria que se produciría. Sería mucho mas practico "almacenar" el conjunto de instrucciones que nos interese y llamarlo cada vez que nos haga falta. A cada uno de dichos conjuntos se les denomina "subrutinas" y se les llama por medio del comando GOSUB seguido de un número de linea (aquel donde comienza la subrutina). Al final de la misma se coloca el comando RETURN, que indica al ordenador donde acaba la subrutina.

Asi pués, cuando la máquina encuentra una sentencia GOSUB continúa la ejecución en el número de linea especificado hasta toparse con RETURN. En este momento, la ejecución continúa en la sentencia que sigue al comando GOSUB correspondiente. De esta manera, el programa anterior se escribiría:

```
10 PRINT "pagina 1"
20 GOSUB 1000
30 PRINT "pagina 2"
40 GOSUB 1000
50 PRINT "pagina 3"
60 GOSUB 1000
.
.
.
.
999 STOP
1000 PRINT AT 21,1;"pulse una tecla para continuar"
1010 PAUSE 0
1020 CLS
1030 RETURN
```

El STOP de la linea 999 es necesario, ya que, de no existir, el programa continuaría ejecutandose en la subrutina y se produciría un error "7 RETURN without GOSUB".

Las subrutinas pueden llamar a su vez a otras o incluso a ellas mismas.

CAPITULO 10

EXPRESIONES MATEMATICAS

El Spectrum puede realizar facilmente todo tipo de operaciones aritméticas (tal como lo haría una calculadora) y para ello se utilizan los signos +, -, * (para multiplicar) y / (para dividir). Tenga en cuenta, a la hora de escribir números decimales, que ha de emplear el punto, y no la coma, para separar la parte decimal de la entera.

El valor resultante de una expresión matemática puede imprimirse directamente o bien utilizarse para asignar un valor a una variable. El siguiente programa calcula el porcentaje que se desee de cualquier número:

```
10 INPUT "numero ";a
20 INPUT "porcentaje ";b
30 LET c=a*b/100
40 PRINT "El ";b;" por ciento de ";a;" es "
50 PRINT : PRINT c
```

En la linea 30 se encuentra la expresión matemática que realiza el cálculo. En este caso, las operaciones se realizan en el mismo orden en que se han escrito; pero no siempre es así. Por ejemplo, pruebe:

```
PRINT 10+20*30
```

el resultado no ha sido el que esperaba; la razón es que la multiplicación se ha efectuado antes que la suma. Esto es debido a que el Spectrum realiza antes las operaciones de multiplicar y dividir y luego las sumas y restas. Se dice que la multiplicación y la división tienen una prioridad mas elevada que la suma y la resta.

Esto puede variarse mediante el empleo de paréntesis; las operaciones entre paréntesis se realizan siempre las primeras y luego las otras (atendiendo a su prioridad).

```
PRINT (10+20)*30
```

Las expresiones pueden incluso utilizarse con las sentencias INPUT, de tal manera que si el ordenador esta esperando un número Vd. puede darle la expresion correspondiente en su lugar.

NOTACION CIENTIFICA

Con este término se denomina a una forma de imprimir números que consiste en escribir un número (decimal o no) y multiplicarlo por 10 elevado a una cantidad. Esto se escribiría:

$$23 \times 10^{3}$$

por ejemplo. En el Spectrum puede también hacerse esto, pero de una forma diferente; así, este mismo numero se escribiria:

```
PRINT 23e3
```

Esto es particularmente útil cuando se trata de operar con cantidades muy grandes. Con este sistema no pueden emplearse expresiones, sino solo valores numéricos

```
PRINT(3+4)e(5*6)
```

sería erroneo.

Hay que tener en cuenta que el Spectrum almacena números con una precisión de 9 1/2 dígitos. Esto significa que las operaciones que se realicen con números de dicha longitud o superiores contendrán errores (aunque posiblemente despreciables) en sus resultados.

CAPITULO 11

FUNCIONES PROPIAS DEL SPECTRUM

Podemos definir una función como un tratamiento especial que se hace con un dato (o conjunto de datos) y que da como resultado otro dato. Esto puede afectar a la clase del dato, su valor numérico, etc.

Al dato que se trata en la función se le denomina "argumento" y al que resulta de aplicar la función se le denomina "resultado". La sintaxis adecuada para utilizar las funciones es la de escribir las mismas colocando el argumento a continuación. Cuando la sentencia que contenga esa función se ejecute, se obtendra el resultado correspondiente.

La primera función que vamos a estudiar es LEN; su argumento es una cadena y su resultado un número que expresa la longitud en caracteres de dicha cadena. Por ejemplo:

```
LEN "ordenador"=9
```

El programa que veremos ahora muestra un uso útil de la funcion LEN.

```
10 INPUT "Mete un texto ";a$
20 FOR n=1 TO LEN a$
30 PRINT a$(n);:PAUSE 7
40 NEXT n
50 GO TO 10
```

Cuando se haya introducido el texto, este no se imprimirá de golpe sino que lo hará letra a letra, como si se tratara de un teletipo o máquina de escribir. Puede alterarse la velocidad de impresión cambiando el valor de la sentencia PAUSE.

Otra función util es STR$; su árgumento es un número y su resultado es una cadena cuyos caracteres serán los del numero introducido

```
STR$ 456="456"
```

En caso de números decimales, la cadena resultante de aplicar STR$ solo contendrá la parte entera del número.

```
STR$ 23.78="23"
```

Una utilidad de esta función sería la escritura de números justificados por la derecha (TAB solo justifica por la izquierda)

```
10 LET cont=0
20 INPUT "Un numero ";a
30 PRINT AT cont,20-LEN STR$ a;a
40 LET cont=cont+1
50 GO TO 20
```

Los numeros se justificarán atendiendo a la columna 20; puede cambiar esto si lo desea (linea 30).

La función VAL es justo la inversa de STR$; su argumento es una cadena y su resultado el valor numérico de la misma (la cadena debe contener unicamente caracteres numéricos a menos que el resultado de VAL pueda ser el nombre de una variable o una expresión). Por ejemplo

```
VAL "34"=34
VAL "8*3"=24
```

La función VAL$ es parecida, aunque menos útil; su argumento es una cadena y su resultado es también una cadena. Su cometido puede ser el de retirar las comillas que puedan sobrar en una cadena

```
VAL$""""spectrum""""="spectrum"
```

La función SGN determina el signo de un número. El resultado es 1 si el número es positivo, 0 si el número tiene ese valor y -1 si se trata de un número negativo.

ABS es una función cuyo argumento es un número y su resultado es el valor absoluto de dicho número.

Veamos un programa que aplica estas dos últimas funciones:

```
10 INPUT "introduzca el saldo ";a
20 IF SGN a=1 THEN PRINT "Tiene ";a;" pesetas"
30 IF SGN a=0 THEN PRINT "Esta en la ruina"
40 IF SGN a=-1 THEN PRINT "Debe ";ABS a;" pesetas"
```

La función INT tiene como argumento un número decimal y su resultado es la parte entera de dicho número.

```
INT 32.5=32
```

esta función siempre redondéa por defecto.

Hasta ahora solo hemos empleado funciones delimitadas por el BASIC del Spectrum. Sin embargo, hay un grupo de comandos que nos permite crear y utilizar nuestras propias funciones; son DEF FN y FN.

DEF FN nos permite crear la función. Se utiliza de la siguiente forma:

```
DEF FN a(x,y)=x*y/2     (por ejemplo)
```

donde "a" es el nombre de la función y "x" r "y" los argumentos.

Para utilizar esta función se emplea FN

```
PRINT FN a(x,y)
```

Tambien puede definirse la función sin necesidad de especificar los argumentos

```
DEF FN a()=x*y/2              PRINT FN a()
```

Hemos colocado solo dos argumentos en el ejemplo anterior, pero se pueden colocar hasta 26 (lógico; los argumentos solo se pueden designar con una letra).

El resultado y los argumentos de una función definida tambien pueden ser cadenas. En ese caso, el formato cambia ligeramente:

```
DEF FN a$(z$)=z$ (2 TO)
```

Una aplicación: dijimos que la función INT siempre redondea por defecto, pero esto puede subsanarse con el empleo de la siguiente función definida:

```
DEF FN a(x)=INT (x+.5)
```

asi redondeara al entero mas proximo.

La función INKEY$ es muy peculiar; no posee argumento y su resultado es una cadena cuyo unico caracter es el de la última tecla pulsada.

```
10 PRINT INKEY$;
20 PAUSE 0
30 GO TO 10
```

Este programa imprimirá los caracteres correspondientes a las teclas que vaya pulsando.

CAPITULO 12

FUNCIONES MATEMATICAS

En este capítulo vamos a tratar de las diversas funciones que permiten al Spectrum la realización de complicados cálculos matemáticos.

La mas importante y versátil es "^", que nos permite hallar la potencia de un número. Como se ve, no se emplea la notación normal para las potencias, pero no parece que esto tenga mucha importancia, dada la facilidad de uso de esta función. Por ejemplo:

```
3^3
```

significa "3 al cubo".

"^" es la operación con la prioridad mas elevada. Es decir, en una expresión matemática, las operaciones con potencias se efectuarán antes que cualquiera de las otras.

También puede emplearse esta función para hallar raices de índice cualquiera. Defina una función así:

```
DEF FN r(x,y)=x^(1/y)
```

siendo x el número e y el índice de la raiz.

El Spectrum dispone de una función específica para hallar la raiz cuadrada de un número: SQR

```
SQR 9=3
```

recuerde que el argumento de SQR nunca debe ser un número negativo. Esto mismo se aplica a "^"; el primer número nunca podrá ser negativo.

La función LN calcula el logaritmo neperiano (o natural) de un número. Recordemos que un logaritmo es el exponente al que hay que elevar un número determinado (llamado "base") para obtener el número que nos interesa. Así:

Log (base 10) de 100=2

Los logaritmos en base 10 se denominan logaritmos vulgares y son los mas corrientemente utilizados. Los logaritmos naturales emplean como base el numero "e".

Así pués, el Spectrum solo calcula directamente los logaritmos neperianos. Si queremos que nos calcule los vulgares, podemos definir la siguiente función:

```
DEF FN l(x)=LN x/LN 10
```

La función EXP calcula el antilogaritmo neperiano de un número dado; es decir, eleva "e" a ese número

```
EXP 10=e^10
```

para hallar el antilogaritmo natural, usaremos

```
10^numero
```

Las funciones trigonométricas (SIN, COS y TAN) calculan el seno, coseno y tangente del ángulo que introduzcamos; también existen funciones que son inversas de las anteriores (ASN, ACS y ATN), es decir, calculan el arcoseno, arcocoseno y arcotangente.

Aclaración: el valor de los ángulos debe ser introducido en radianes; si queremos introducirlo en grados, usemos el siguiente programa de conversión:

```
10 INPUT "Angulo? ";a
20 LET ang=a/180*PI: REM ang es el angulo en radianes
```

Asimismo, las funciones inversas nos darán el resultado en radianes. Para pasarlo a grados, teclear:

```
10 REM rad es el angulo en radianes
20 LET ang=rad/PI*180: REM ang es el resultado
```

PI es palabra clave en el Spectrum; se obtiene pasando a modo extendido y pulsando luego "M".

CAPITULO 13

NUMEROS ALEATORIOS

En muchos programas (sobre todo de juegos) es muy util disponer de un generador de números aleatorios. El Spectrum posee para este cometido la función RND, la cual genera un número diferente, comprendido entre 0 y 1, cada vez que se activa. Pruebelo de esta manera:

```
10 FOR n=1 TO 20
20 PRINT RND
30 NEXT n
```

Como puede verse RND, a pesar de ser una función, no posee argumento.

Si ha ejecutado el programa anterior, habra podido comprobar que el numero que se genera, o es 0 o un número decimal menor que 1. Esto no es que sea muy útil, ya que lo que nos interesara mas a menudo es la creación de números enteros. Para ello, si queremos generar números aleatorios comprendidos entre m y n, (ambos inclusive) haremos:

```
10 FOR a=1 TO 20
20 PRINT INT (RND*(n-m+1))+m
30 NEXT a
```

En realidad, los números generados no son verdaderamente aleatorios, sino que forman parte de una sucesión de 65536 números. Sin embargo, esta sucesion se ha estructurado sin ninguna norma especifica y eso, unido a su longitud, simula cierta aleatoriedad. Podemos decir por ello que RND es una función "pseudoaleatoria".

El comando RANDOMIZE nos permite hacer que RND comience a contar desde el lugar de la sucesión que nos interese. Compruebelo:

```
10 RANDOMIZE 1
20 FOR n=1 TO 15
30 PRINT INT (RND*10)
40 NEXT n: PAUSE 100
50 CLS: GO TO 10
```

Verá como la sucesión de números es siempre la misma.

Si se utiliza RANDOMIZE (o RANDOMIZE 0) el valor asignado para la sucesión se basará en el tiempo que el ordenador lleve funcionando. Compruébelo con el siguiente programa:

```
10 RANDOMIZE
20 PRINT PEEK 23670 + 256*PEEK 23671
30 GO TO 10
```

Los números que aparezcan, aunque crecientes, se diferencian poco el uno del otro. Sin embargo, cuando el ordenador le pregunte "scroll?" tarde mas o menos en pulsar la tecla para continuar viendo la sucesión; cuanto mas tiempo tarde, mas diferencia habra entre el último número escrito y el que aparezca inmediatamente después de pulsar la tecla.

CAPITULO 14

OPERADORES LOGICOS

En el capítulo 8 vimos como el ordenador podía tomar decisiones mediante el conjunto de instrucciones IF - THEN. Estas decisiones se tomaban en función de que unas condiciones fueran verdaderas o falsas, y para establecerlas utilizábamos los signos <, >, <=, >=, = o <>. A estos símbolos se les denomina relaciones. Las relaciones no tienen por qué emplearse por separado; pueden combinarse varias de ellas por medio de los operadores AND, OR y NOT (todos ellos palabras clave en el Spectrum).

Supongamos que queremos relacionar dos condiciones por medio de operadores logicos. Estos se utilizan de la siguiente manera:

condición1 AND condición2	La relación es verdadera si ambas condiciones lo són.
condición1 OR condición2	La relación es verdadera si una de las condiciones lo es.
NOT condición	Si la condición es falsa, la relación es verdadera y viceversa.

La prioridad en la ejecución de estos operadores es mas baja que la de las relaciones y las operaciones.

Ejemplos:

```
10 PRINT "Meta 3 numeros distintos
20 INPUT a;" ";b;" ";c
30 IF a=b OR b=c OR a=c THEN PRINT "no vale": GO TO 20
```

Debemos tener en cuenta que las condiciones tienen un valor numérico; si son verdaderas tendran valor 1 y si son falsas, valor 0. Puede constatarlo tecleando:

```
PRINT 4=4,4<>4
```

se imprimirán los numeros 1 y 0, en este orden.

Las operaciones con AND, OR y NOT tambien tienen valor numérico. Así:

a AND b   vale a si b es verdadera y 0 en caso contrario.

a OR b   vale 1 si b es verdadera y a en caso contrario.

NOT a   vale 0 si a es verdadera y 1 en caso contrario.

Recuerde que 1 equivale a "verdadero" y 0 a "falso"

Veamos una aplicación que nos aclarará lo anterior. Supongamos qe queremos que un programa salte a una linea determinada segun una opción que escojamos. Podemos hacerlo:

```
10 PRINT "Escoje opcion"
20 PRINT TAB 10;"1 - Salto a linea 100"
30 PRINT : PRINT TAB 10;"2 - Salto a linea 300"
40 PRINT : PRINT TAB 10;"3 - Salto a linea 800"
50 PAUSE 0: LET a$=INKEY$ : CLS
60 IF a$="1" THEN GOTO 100
70 IF a$="2" THEN GOTO 300
80 IF a$="3" THEN GOTO 800
100 PRINT "Linea 100" : STOP
300 PRINT "Linea 300" : STOP
800 PRINT "LINEA 800" : STOP
```

Sin embargo, el programa podría escribirse mucho mas sencillamente de la siguiente manera.

```
10 PRINT "Escoje opcion"
20 PRINT TAB 10;"1 - Salto a linea 100"
30 PRINT : PRINT TAB 10;"2 - Salto a linea 300"
40 PRINT : PRINT TAB 10;"3 - Salto a linea 800"
50 GOTO (100 AND INKEY$="1")+(300 AND  INKEY$="2")+(800
   + INKEY$="3")
100 PRINT "Linea 100" : STOP
300 PRINT "Linea 300" : STOP
800 PRINT "LINEA 800" : STOP
```

Este programa ocupará mucha menos memoria que el anterior y tendrá la ventaja de que solo funcionará si se pulsa una de las tres teclas especificadas, ahorrandonos así la linea de control necesaria. Acostúmbrese a emplear esta técnica en lugar de las sentencias IF - THEN.

Por supuesto, podemos utilizar ésto tanto con cadenas como con números (y con variables de los dos tipos).

CAPITULO 15

EL JUEGO DE CARACTERES

Como ya sabemos, el Spectrum (y todos los ordenadores) solo pueden manejar valores numéricos. Eso significa que tendremos que codificar todas las instrucciones y los caracteres que empleemos para escribirlas.

Se denominan caracteres a todas las letras, números, símbolos y (en el Spectrum) palabras clave de BASIC que utilicemos para escribir nuestros programas o los que utilice el ordenador para entregarnos datos por pantalla o impresora. Hay 256 caracteres, numerados del 0 al 255 y al conjunto de todos ellos se le denomina "juego de caracteres".

Existen dos funciones que se utilizan para manejar directamente el juego de caracteres. Son CODE y CHR$.

El argumento de CODE es una cadena de un solo caracter y su resultado el código correspondiente a ese caracter.

El argumento de CHR$ es un número de 0 a 255 y proporciona una cadena cuyo único caracter es el correspondiente al número introducido.

Al final de este libro se encuentra la lista del juego de caracteres, colocados por orden creciente de sus códigos. Si queremos imprimir el juego en pantalla, empleemos el siguiente programa:

```
10 FOR n=32 TO 255
20 PRINT "(";n;")";CHR$ n
30 NEXT n
```

Los primeros 31 codigos no corresponden a ningún caracter que se pueda imprimir, por eso los excluimos. Delante de cada caracter se imprime entre paréntesis el código correspondiente. No le cabrán todos juntos en la pantalla, así que vaya pulsando teclas cada vez que se llene la misma.

Los caracteres cuyos codigos estan comprendidos entre 32 y 126 estan tomados de un juego de caracteres standard muy usado en los EEUU.

Este código se denomina ASCII (American Standard Codes for Information Interchange). Los caracteres incluidos en el juego del Spectrum tienen el mismo código que el que tienen en el ASCII. Hay una excepción: el signo £.

El caracter 127 es el simbolo de Copyright. Desde el 128 hasta el 143 se encuentran los símbolos gráficos preprogramados en el Spectrum y que se encuentran en las teclas numéricas de la 1 a la 8. Desde el 144 al 164 encontramos los 21 símbolos gráficos que podemos programar segun queramos, aunque aquí aparecen en forma de letras mayúsculas; ésto se debe a que el ordenador, al ser conectado, configura estos gráficos como tales letras. Mas adelante, y en este mismo capítulo, estudiaremos la manera de programar estos caracteres.

Del numero 165 en adelante se encuentran los "tokens" o palabras clave del Spectrum. Tanto éstas como los símbolos gráficos no forman parte del codigo ASCII sino que son propios del Spectrum.

Como habíamos dicho, los primeros 32 caracteres (del 0 al 31) no han sido incluídos en la relación, ya que no producen nada que pueda imprimirse. Se les denomina "caracteres de control".

Uno de estos caracteres puede ser de gran utilidad; se trata de CHR$ 8. Este caracter hace retroceder la posición de impresión una posición de caracter hacia la izquierda. Pruebe:

```
PRINT "perro";CHR$ 8;"a"
```

Para terminar, ejecute el siguiente programa:

```
10 PRINT CODE INKEY$;" ";
20 PAUSE 0
30 GO TO 10
```

El primer número que aparecerá sera un cero; efectivamente, este es el código correspondiente a "ninguna tecla". Ahora entreténgase pulsando las teclas de cursor, los modos, EDIT, DELETE y ENTER. Verá como a todas corresponde un número. Esta es la manera en que el Spectrum comprende todas estas operaciones.

Aprovechamos para indicarle que cuando utilice las condiciones <, >, >= o <= estas atenderán, no al auténtico orden alfabético sino al orden en que se encuentren las letras en el ASCII.

Veamos ahora unas cuantas cadenas en el orden en que serían colocadas por el Spectrum:

CHR$ 8 + "hola"
" zapato"
"120"
"Suceso"
"aqui"

Vamos ahora a ver como se definen los caracteres gráficos (también llamados "GDU") que nos interesen. Para entender ésto, vamos a ver primero la manera en que el Spectrum compone los caracteres.

Cada "posicion de caracter" está compuesta por una retícula de 8 x 8 cuadraditos, como si fuera una hoja de papel cuadriculado. A cada uno de estos cuadraditos se le denomina "pixel" y al estar oscurecidos o no configuran el caracter que nos interesa.

Cada linea de 8 cuadraditos se almacena en una posición determinada de memoria, por orden de arriba a abajo. Dado que cada linea representa un byte, podemos pensar en ella como un numero binario de ocho dígitos. Los ceros de dicho número corresponderán a los cuadraditos claros y los unos a los oscuros. Para ilustrar ésto, ejecute el siguiente programa:

```
10 PLOT 95,80: DRAW 0,64: DRAW 64,0: DRAW 0,-64: DRAW
-64,0
15 PRINT AT 4,12;"  ████ "
20 PRINT AT 6,12;" ████  "
30 PRINT AT 7,12;" █   █ "
40 PRINT AT 8,12;" █   █ "
50 PRINT AT 9,12;" █   █ "
60 PRINT AT 10,12;" █   █ "
70 PRINT AT 4,8;60 ;AT 5,8;0;AT 6,8;120;AT 7,8;68;AT
8,8;68;AT 9,8;68;AT 10,8;68;AT 11,8;0
80 FOR n=1 TO 64 STEP 8
90 PLOT 95,n+79: DRAW 64,0
100 PLOT n+94,80: DRAW 0,64
110 NEXT n
```

No haga caso a los comandos que no conozca; limítese a introducirlo.

Los cuadrados negros han de introducirse activando el modo gráfico y pulsando CAPS SHIFT y 8 simultaneamente para imprimir cada cuadrado. Recuerde volver al modo "L" cuando haya terminado.

Al ejecutar el programa le aparecera el caracter "ñ" definido sobre la correspondiente rejilla de 8 x 8. Los números que aparecen a la izquierda son los valores correspondientes a cada número que tendremos que introducir en la memoria.

Para hallar estos valores hemos hecho lo siguiente:

1.- Escribir los números binarios correspondientes, basandonos en el caracter que queramos definir. En este caso:

```
00111100
00000000
01111000
01000100
01000100
01000100
01000100
00000000
```

Si sustituimos los unos por cuadrados negros y los ceros por espacios, observaremos como coincide la configuración resultante con la del caracter.

2.- Pasar los numeros binarios a números decimales. Hay una función en el Spectrum que pasa directamente los números binarios a decimales: BIN. Hagamos:

```
10 INPUT A
20 PRINT BIN A
30 GO TO 10
```

Introduzcamos los números binarios resultantes del apartado 1 y tendremos sus valores decimales. Como se ve, coinciden con los que aparecen impresos en pantalla en el programa.

Lo siguiente que debemos hacer es introducir estos números en la memoria del ordenador. Supongamos que vamos a definir este caracter en el gráfico "A" es decir, que al pulsar la "A" en modo gráfico nos aparecerá el caracter. La dirección del primer byte es USR "a", la siguiente USR "a"+1, la siguiente USR "a"+2 y asi sucesivamente hasta llegar a USR "a"+7.

USR es una función cuyo argumento es un caracter de la "a" a la "u" y su resultado es la dirección del primer byte del correspondiente gráfico definido por el usuario. Hay otro uso para USR, que veremos en un próximo capitulo.

Para introducir los números en la memoria usaremos el siguiente programa:

```
10 FOR n=0 TO 7
20 READ a: POKE USR "a"+n,a
30 NEXT n
40 DATA 60,0,120,68,68,68,68,0
```

POKE es un comando que introduce directamente un número en la memoria del ordenador. Ese número debe estar comprendido entre 0 y 255. Su inverso es la función PEEK, que permite leer una posición de memoria determinada. Hablaremos mas sobre ésto en el capítulo dedicado a la memoria.

CAPITULO 16

LOS COLORES DEL SPECTRUM

El Spectrum es capaz de producir imagenes en color. Se puede escoger entre 8 colores: Negro, Azul oscuro, Rojo, Magenta, Verde, Azul Claro, Amarillo y Blanco. Como se ve, hemos descrito los colores dependiendo de su brillo, del mas oscuro al mas claro; asimismo, para designar estos colores se emplea un número del 0 al 7. Pruebe:

```
10 FOR n=0 TO 7
20 PRINT PAPER n;"
30 NEXT n
```

tendremos 8 franjas de color diferente. Como se ve 0 corresponde a negro (mas oscuro) y 7 al blanco (el mas claro).

Todos estos colores pueden emplearse simultaneamente en la pantalla; la unica limitación que existe es que solo pueden definirse dos colores por cada posicion de carácter : uno para l parte oscura (INK) y otro para la parte clara (PAPER). Para comprender mejor ésto, revise en el capitulo 15 la parte dedicada a definición de caracteres.

Existen otros dos comandos de color, BRIGHT y FLASH, que pueden utilizarse conjuntamente con INK y PAPER. El primero indica "brillo especial" y se activa con BRIGHT 1. El segundo indica "parpadeo" y se activa con FLASH 1. Estos dos comandos se desactivan con BRIGHT 0 y FLASH 0, respectivamente.

Como en el caso de PAPER e INK, solo puede definirse un valor de BRIGHT y FLASH para cada posición de caracter. La combinación de un valor de PAPER, otro de INK, otro de BRIGHT y otro de FLASH se denomina "atributo" de esa posición de carácter.

Hay otro comando que controla el color del contorno de la pantalla (donde no puede escribirse) y el que tiene el fondo de las dos lineas inferiores (las empleadas para informes). Se llama BORDER y se utiliza de la siguiente forma (ejemplo):

```
10 FOR n=0 TO 7
20 BORDER n: PAUSE 20
30 NEXT n
```

Al conectar el ordenador, todas las posiciones tienen valor de INK, BRIGHT y FLASH igual a 0 y PAPER y BORDER igual a 7 (es decir, todo se imprime en negro sobre fondo blanco y sin parpadeo).

Pueden emplearse estos comandos para cambiar los atributos de toda la pantalla; ésto funcionara directamente con todos menos con PAPER. Pruebe:

```
10 BORDER 2: PAPER 2: INK 7
20 PRINT "Fondo rojo, letras en blanco"
30 PAUSE 0: PRINT "Ahora toda la pantalla"
```

Al ejecutar el programa, el primer mensaje aparecerá con los colores que ha definido, pero el resto de la pantalla seguirá de color blanco. Pulse una tecla y entonces toda la pantalla tendrá los atributos definidos en la linea 10.

Si desea que los colores de la pantalla completa cambien desde el primer momento, utilice la siguiente linea al principio del programa:

```
BORDER 2: PAPER 2: INK 7: CLS
```

Los valores normales de INK y PAPER están comprendidos entre 0 y 7, pero pueden utilizarse también los números 8 y 9. El número 8 puede emplearse con todos los comandos de color (menos con BORDER). Su efecto es que la sentencia PRINT que sigue a aquella en la que se ha aplicado imprimirá la información en el mismo color. El numero 9 sólo puede emplearse con PAPER e INK e indica que el papel (o la tinta) han de contrastar; es decir, que si se emplea un papel claro, la tinta será oscura y viceversa.

Cualquier utilización erronea de números para estos comandos dará origen al error "K Invalid Colour".

Los atributos de las dos líneas inferiores se controlan normalmente por medio de BORDER e INK 9; no se puede utilizar en ellos PAPER, BRIGHT ni FLASH.

Existen otros dos comandos, INVERSE y OVER que, aunque no modifican los atributos alteran la impresión. Sus valores pueden ser 0 o 1 (igual que en el caso de BRIGHT y FLASH). INVERSE 1 realiza el efecto de cambiar el papel por la tinta; es decir, los cuadraditos oscuros se vuelven claros y viceversa.

OVER 1 permite la sobreimpresión de dos caracteres en una misma posición de caracter; de no emplearse OVER, el nuevo caracter simplemente borraría al antiguo. Este programa utiliza OVER para imprimir la "ñ":

```
10 OVER 1
20 PRINT "nin";CHR$ 8;" o"
```

(Recuerde que CHR$ 8 servía para retroceder la posición de impresión)

Los comandos INK, PAPER, etc., pueden utilizarse como complementarios de las sentencias PRINT, con el fin de que actuen sólo con los elementos de dichas sentencias.

```
10 PRINT PAPER 1;INK 7;"esto en tinta blanca, papel azul"
20 PRINT "esto en el papel y tinta definido antes"
```

Podemos emplear esta cualidad para introducir atributos en la parte inferior de la pantalla. Por ejemplo, podemos utilizar INK y PAPER en una sentencia INPUT

```
INPUT PAPER 2; INK 7;"color ";a
```

Se puede tambien escribir en las dos ultimas lineas de la pantalla (y emplear comandos de color) de la siguiente forma:

```
PRINT # 1; PAPER 4; INK 0; AT 0,1;"PULSE UNA TECLA PARA
CONTINUAR": PAUSE 0
```

Recuerde en este caso que la aparición de cualquier mensaje o cursor INPUT borrará lo que hemos escrito en estas lineas.

Podemos omitir el uso de los comandos de color y establecer los atributos por medio de caracteres de control. Recuerde que en capitulo anterior hablábamos de caracteres que no podían imprimirse. Los caracteres de control que manejan el color son:

CHR$ 16 equivale a INK
CHR$ 17 equivale a PAPER
CHR$ 18 equivale a FLASH
CHR$ 19 equivale a BRIGHT
CHR$ 20 equivale a INVERSE
CHR$ 21 equivale a OVER

Asi pues

```
CHR$ 17 + CHR$ 4
```

equivale a

```
PAPER 4
```

Estos caracteres no tienen por que ser introducidos necesariamente empleando CHR$. Pueden introducirse directamente en Modo Extendido utilizando las teclas numéricas. Las reglas para ésto son las siguientes (siempre en Modo Extendido):

Los numeros 0 al 7 controlan el color del papel.

Si se utilizan con CAPS SHIFT determinan el color de la tinta

Pulsando 8 y 9 se controla el brillo (9 lo conecta, 8 lo desconecta).

Pulsando 8 y 9 con CAPS SHIFT se actua sobre el FLASH (9 activar, 8 desactivar).

Los caracteres de control funcionarán en un listado BASIC mientras no se cambien. Para comprobar ésto haga lo siguiente:

```
10 PRINT "prueba de caracteres de control"
20 PRINT "continuacion"
```

En la linea 10, despues de las comillas, introduzca un caracter de control (por ejemplo, pulsando Modo Extendido, apriete el numero 6) y la frase aparecera afectada por el caracter introducido. Al introducir la siguiente linea, esta quedará tambien afectada. Para eliminar el efecto del caracter de control deberá introducir otro al final de lo que quiera imprimir.

Ahora bién, si corre el programa verá que, aunque el listado ha resultado afectado, no ha sido así con el texto impreso en la linea 20. De todas formas, tenga cuidado al utilizar los caracteres de control en listados, para evitar confusiones en el mismo.

Aunque los colores a utilizar no sean mas que 8 podemos simular más tonalidades empleando un gráfico definido por el usuario. El programa puede ser como sigue:

```
10 FOR n=0 TO 7
20 READ a: POKE USR "a"+n,a
30 NEXT n
40 INPUT a,b
50 IF a>8 OR b>8 THEN GO TO 40
60 PRINT PAPER a;INK b;"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
70 GO TO 40
80 DATA 85,170,85,170,85,170,85,170
```

(Las letras "A" de la linea 60 han de ser entradas en Modo Gráfico)

El programa imprimirá una serie de lineas de diferentes colores (hasta 64) segun sea la combinación de colores que introduzca por medio de la sentencia INPUT de la linea 40. Por supuesto que si introduce dos números iguales la franja será del color correspondiente.

En el capítulo dedicado al fichero de pantalla hablaremos mas del uso de los colores.

Para terminar, diremos que existe una función llamada POINT, cuyo argumento es una coordenada de pixel determinada y cuyo resultado es 1 ó 0 (dependiendo si el pixel correspondiente es de tinta o de papel, respectivamente). En el capitulo dedicado a los dibujos en alta resolución hablaremos de las coordenadas de pixel.

CAPITULO 17

GRAFICOS EN ALTA RESOLUCION

Recordemos que en la pantalla teníamos disponibles 32 x 22 (704) posiciones de caracter. Como cada una de esas posiciones disponía de 8 x 8 (64) cuadraditos o "pixels", resulta que tenemos disponibles 256 x 176 pixels, que podemos oscurecer o dejar en blanco a voluntad.

Esto nos permite la realización de dibujos en alta resolución. Los comandos que permiten este tipo de dibujo utilizan como argumento las coordenadas de la pantalla en alta resolución. Los extremos de estas coordenadas son 0,0 para la esquina inferior izquierda, 255,0 para la inferior derecha, 0,175 para la superior izquierda y 255,175 para la superior derecha.

El comando PLOT define un punto en las coordenadas que le especifiquemos

```
10 FOR n=1 TO 200 STEP 5
20 PLOT n,50
30 NEXT n
```

Este otro programa traza la gráfica del seno de los angulos comprendidos entre 0 y 360 grados.

```
10 FOR x=0 TO 255
20 PLOT x,88 + 80*SIN (n/128*PI)
30 NEXT x
```

(Recuerde que tanto SIN como PI son palabras clave)

La sentencia DRAW traza una linea recta

```
PLOT 0,100: DRAW 100,0
```

trazará una linea horizontal partiendo del punto 0,100 de 100 pixels de longitud. Si ahora hacemos:

```
DRAW 30,30
```

veremos que la nueva linea trazada tiene su origen en el extremo final de la anterior.

El punto desde donde actúa una sentencia DRAW se denomina "posición PLOT" y cambia con cada ejecución de una sentencia DRAW. Pruebe lo siguiente:

```
10 PLOT 50,50: DRAW 30,0
20 DRAW 0,30
30 DRAW -30,0
40 DRAW 0,-30
```

obtedremos un cuadrado de lado 30 pixels. Pruebe a introducir PLOT 50,50 al principio de las lineas 20, 30 y 40 y vea como cambia la ejecución del programa.

Observe que se pueden emplear números negativos en una sentencia DRAW. Nunca pueden emplearse con PLOT.

RUN, CLEAR, NEW y CLS colocan la posición PLOT en el punto 0,0.

Se pueden utilizar comandos de color en sentencias DRAW, aunque se deben tener en cuenta las limitaciones para la resolución de los colores en la pantalla. Pruebe lo siguiente:

```
10 FOR n=1 TO 200 STEP 10
20 PLOT n,50: DRAW INK INT (RND*8);0,50
30 NEXT n
40 PLOT 0,80: DRAW INK 6;200,0
```

Como vemos, la linea horizontal ha alterado el color de las lineas verticales en la posición de caracter por donde las ha atravesado.

Puede emplearse DRAW con un tercer número para trazar arcos de circunferencia. El número puede considerarse como la fracción de círculo que se quiera trazar. Pruebe:

```
10 PLOT 80,80: DRAW 50,50,PI
```

Dibujará una semicircunferencia.

Pueden obtenerse efectos curiosos empleando valores grandes para el tercer número. Borre la pantalla y teclee:

```
PLOT 80,80: DRAW 50,50,350
```

El comando CIRCLE dibuja un círculo. Se emplea así:

```
CIRCLE x,y,r
```

siendo x e y las coordenadas del centro del círculo y r el radio del mismo (en pixels).

Ya hemos visto que se pueden emplear los comandos PAPER e INK con una sentencia PLOT o DRAW. También pueden emplearse OVER e INVERSE, aunque los efectos conseguidos serán diferentes a los que se obtienen normalmente.

PLOT INVERSE 1 dibujará un "punto borrador", es decir, que sí hacemos PLOT INVERSE 1 en un punto previamente definido éste se borrará; para ser mas exacto, esta sentencia pone de color PAPER el punto a donde se aplique.

PLOT OVER 1 posee un efecto similar al anterior, pero mas completo: si se aplica sobre un pixel "PAPER" lo convertirá en "INK" y viceversa.

Veamos un ejemplo. Teclee:

```
10 PRINT AT 11,0;"Vamos a tachar esta linea"
20 PLOT 0,84: DRAW 210,0
30 PAUSE 200
40 PLOT 0,84: DRAW INVERSE 1; 210,0
```

Aunque la linea que tachaba la escritura ha sido borrada, queda el rastro de la misma. Sin embargo si escribimos el programa así:

```
10 PRINT AT 11,0;"Vamos a tachar esta linea"
20 PLOT 0,84: DRAW OVER 1; 210,0
30 PAUSE 200
40 PLOT 0,84: DRAW OVER 1; 210,0
```

La linea desaparecerá sin dejar rastro.

CAPITULO 18

SONIDO: USO DE BEEP

El Spectrum puede producir sonidos que sirvan para complementar sus programas. Para ello se dispone del comando BEEP, que se utiliza de la siguiente forma:

```
BEEP duracion, tono
```

La duración se expresa en segundos y el tono en semitonos. Los valores extremos que se pueden utilizar son de 0 a 10 para la duración y de -60 a 69 para el tono. Los valores pueden ser números fraccionarios.

Pruebe este programa:

```
10 FOR n=-10 TO 70
20 BEEP .5,n
30 NEXT n
```

tocará notas cada vez mas altas y terminará parandose con un mensaje de error "Integer out of range".

El valor 0 es el "do" central de la escala musical. El resto de los números siguen la misma escala que las teclas consecutivas de un piano. Es decir, 1 es RE bemol, 2 es RE, 3 MI bemol, etc.

Desgraciadamente su Spectrum solo puede tocar una nota de cada vez, de forma que los acordes están prohibidos. Es posible simularlos con rutinas en código máquina que en realidad se limitan a tocar dos notas una detras de la otra, pero lo hace tan rapido que parece que se oyen juntas.

Puede simularse algunos de estos efectos empleando BASIC. Pruebe:

```
10 FOR n=4 TO 32 STEP 4: FOR m=n TO 4 STEP -1
20 FOR p=32 TO n-64 STEP -10: BEEP .01,m
30 NEXT p: NEXT m: NEXT n
```

CAPITULO 19

LA MEMORIA

Ya dijimos al principio de este manual que el microprocesador con que esta equipado el Spectrum solo entiende las instrucciones si se le envían en forma de números binarios. Cada 0 o 1 de cada uno de estos números binarios se denomina "bit" y consecuentemente solo puede tener dos valores: 0 o 1.

Cada número binario que se procesa esta compuesto de 8 bits y se le denomina "byte" (se pronuncia "bait") u "octeto". El mayor número que se puede almacenar en un byte es 255 (es decir 11111111 binario). Podemos decir que un byte es la menor unidad de memoria.

Cada uno de los lugares de la memoria del ordenador en donde se puede almacenar un byte se denomina "posición de memoria" o "dirección" El Spectrum posee un total de 65536 de estas posiciones, numeradas de la 0 a la 65535.

Las posiciones 0 a 16384 constituyen la ROM (abreviatura de Read Only Memory - Memoria de solo lectura). En estas posiciones se almacena el "sistema operativo" (programa escrito en código máquina que sirve para hacer comprender al ordenador las instrucciones), el lenguaje BASIC y el juego de caracteres que se usa para imprimir los textos en la pantalla. Como su nombre indica, esta memoria no puede ser borrada ni se puede escribir en ella; la información contenida en la misma esta disponible en cuanto se conecta el ordenador.

Las posiciones de memoria 16384 a 65535 constituyen la memoria RAM (Random Access Memory - Memoria de acceso aleatorio). En esta memoria se puede escribir, leer o borrar lo que interese. Toda la información en ella contenida se borra al desconectar el ordenador.

Cuando se escribe o ejecuta un programa en BASIC, el ordenador va asignando valores comprendidos entre 0 y 255 a los bytes de la RAM. Se pueden introducir valores directamente en las distintas posiciones de memoria utilizando el comando POKE de la siguiente forma:

```
POKE direccion, contenido
```

La dirección puede estar comprendida entre 16384 y 65535 (la RAM). Si utilizamos POKE en la ROM el ordenador simplemente no nos hara ni caso.

Para leer una dirección se emplea la función PEEK. Pruebe:

```
POKE 40000,30
```

y luego

```
PRINT PEEK 40000
```

Como se ve, el uso de POKE es realmente útil cuando se trata de introducir datos directamente en la memoria, aunque también puede ser algo muy peligroso; puede alterarse substancialmente el programa que esté en ese momento en memoria.

A partir de la dirección 16384 hasta la 23296 se almacena la imagen para la pantalla de televisión. Es lo que se llama el "fichero de pantalla". Los primeros 6144 bytes almacenan la imagen propiamente dicha y los 768 restantes los colores de la misma. Hablaremos mas extensamente de esto en el capítulo dedicado al fichero de pantalla.

Desde la dirección 23296 a la 23552 se encuentra el "buffer" o memoria intermedia de la impresora. En ella se guardan los caracteres destinados a ser impresos. Cuando se ha llenado, su contenido se envía a la impresora.

Desde la 23552 a la 23734 se almacenan las denominadas "variables del sistema". Son valores que el propio ordenador actualiza según sus necesidades. Se hablará de las funciones y utilidades de estas variables en el capítulo correspondiente.

En la dirección 23734 comienza una zona denominada "Mapas de Microdrive". Como su nombre indica, solo estará ocupada si está conectado ese periférico. A continuación de esta zona viene otra destinada a retener información sobre el último canal utilizado. Los canales más importantes son el "K" (teclado), "S" (pantalla) y "P" (impresora). La dirección donde comienza esta zona está almacenada en la variable del sistema CHANS.

Los programas en BASIC comienzan normalmente en la dirección 23755.

Primero se almacena el programa en sí, luego las variables (a partir de la dirección especificada en la variable del sistema VARS), después la linea de programa que se está editando y por último una serie de datos. Por ejemplo, los números con los que se está calculando o las direcciones a las que hay que volver en caso de GOSUB.

La zona de BASIC termina en la dirección contenida en la variable RAMTOP. Puede alterarse este valor utilizando el comando CLEAR de la siguiente manera:

```
CLEAR nueva direccion de RAMTOP
```

Esta sentencia tiene mas efectos aparte del ya descrito de cambiar la RAMTOP; borra todas las variables, el fichero de pantalla y restablece la posición PLOT a las coordenadas 0,0. Tambien traslada la pila de GOSUB (donde se almacenan las direcciones de vuelta de las subrutinas).

Los últimos 168 bytes de la memoria contienen los datos correspondientes a los G.D.U. (graficos programables).

La sentencia CLEAR permite reservar espacio para programas en código máquina, reduciendo el area del BASIC, o por el contrario sumar a la misma la zona de gráficos definidos por el usuario.

Como dijimos, en un byte solo se puede almacenar un número igual o menor de 255. Pero el ordenador también puede almacenar números hasta 65535 utilizando 2 bytes. El número se descompone en otros dos menores de 255. Para ver como sucede ésto, ejecute el siguiente programa:

```
10 INPUT "Numero de 0 a 65535";a
20 RANDOMIZE a
30 PRINT PEEK 23670,PEEK 23671
```

Cuando haya introducido el número, aparecerán en pantalla dos números; al primero se le llama "byte menos significativo" y al segundo "byte mas significativo". Como se ve, se han almacenado estos números en dos posiciones de memoria correlativas, el menos significativo el primero. Para reconstruir el número haga:

```
PRINT PEEK 23670+256*PEEK 23671
```

Todos los números se almacenan en la memoria en este orden, con una excepción: los números de linea del BASIC. Teclee lo siguiente:

```
10 REM linea 0
```

y luego:

```
PRINT PEEK 23756+256*PEEK 23755
```

aparecerá el numero 10. Ahora haga:

```
POKE 23755,0: POKE 23756,0
```

!!sorpresa !! el listado se vera así ahora:

```
0 REM linea 0
```

trate de editar esta linea.

Para restablecer haga los siguiente:

```
POKE 23756,1
```

Muchas variables del sistema emplean este método para introducir números. Lo veremos en el capítulo correspondiente.

CAPITULO 20

MEDIOS DE ALMACENAMIENTO EXTERNO

Como dijimos en el capítulo dedicado a la memoria, la información contenida en la memoria RAM se borrará cada vez que se desconecte el ordenador. Como no podemos tener el Spectrum conectado indefinidamente, esto significará la pérdida de los programas y datos.

Para poder utilizar de nuevo la información tendríamos dos medios; el primero, volver a introducirla toda en el ordenador, lo que no es una solución muy práctica...la segunda (que es la que se emplea) consiste en almacenarla en algún dispositivo ajeno al propio ordenador, antes de desconectar éste; de esta forma, si queremos recuperar la información, solo tendremos que cargarla en el equipo desde ese dispositivo.

Todos los sistemas de almacenamiento se basan en soportes magnéticos: cintas o discos. El más corrientemente utilizado en el Spectrum (por su economía) es la cinta de cassette. El grabador que se utilice debe cumplir ciertas condiciones:

- no ser estereofonico

- poseer una buena respuesta de frecuencia entre 900 y 1100 Hertzios

- Tener correctamente ajustada la cabeza de grabación-reproducción.

- Tener un nivel de salida lo suficientemente fuerte como para que los datos puedan ser recibidos por el ordenador.

si alguna de estas características le son desconocidas, consulte a un especialista. Para conectar el cassette, siga las instrucciones contenidas en la Introducción de este libro.

Los comandos utilizados para la carga y grabación son los siguientes:

1 - SAVE Se usa así: SAVE "nombre del programa"

Este comando graba en la cinta el programa en BASIC que hay en la memoria. Una vez introducido, aparece el mensaje "start tape and press any key". Retire entonces la clavija EAR, arranque el cassette (puesto en modo "grabación") y pulse cualquier tecla. Aparecerá una secuencia de dos segundos de duración de rayas iguales de color rojo y azul y a continuación otra muy corta de rayas amarillas y negras. Esto nos indica que el ordenado ha introducido un corto bloque de información que le servirá mas tarde de referencia para cargar el programa desde la cinta. Este bloque contiene el tipo de información (BASIC, código máquina o matriz), la longitud del bloque y la linea de autoejecución (si existe) o la dirección de comienzo (si se trata de código máquina). A este bloque se le llama "cabecero".

Despues de ésto, aparecerá otra secuencia de un segundo de rayas azules y rojas y después otra configuración de rayas amarillas y negras. Esta última corresponde al programa propiamente dicho.

Si se desea grabar el programa de manera que se autoejecute se utiliza el comando:

SAVE "nombre" LINE linea de autoejecución

(LINE es palabra clave)

Una vez cargado, el programa comenzará a ejecutarse a partir de la linea que hemos indicado.

Si lo que se desea grabar es código máquina, haremos:

SAVE "nombre" CODE dirección, longitud

"dirección" es la posición en la memoria del primer byte que se desea grabar. "longitud" es la cantidad de bytes a grabar desde la "dirección".

SAVE tambien permite grabar matrices, que pueden ser utilizadas de esta forma como archivos externos de datos. Se hace así:

SAVE "nombre" DATA "nombre de la matriz" ()

El nombre de la matriz debe estar compuesto por una letra (en caso de que sea una matriz numérica) o por una letra y el signo "$" (matriz alfanumérica). Ese es el nombre que tiene la matriz en el programa que la contiene.

Un caso especial de utilización de SAVE es:

```
SAVE "nombre" SCREEN$
```

que almacenará la imagen que se encuentre en ese momento en la pantalla de televisión. En realidad equivale a:

```
SAVE "nombre" CODE 16384,6912
```

2 - VERIFY

Este comando se usa de la misma forma que SAVE. Su función es verificar la grabación que se acaba de realizar. Para ello se rebobina la cinta hasta el inicio del bloque que hemos grabado, se vuelve a conectar la clavija EAR y se introduce el comando de una de estas tres maneras:

```
VERIFY "nombre" (programas BASIC)
VERIFY "nombre" CODE (código máquina)
VERIFY "nombre" DATA "nombre de la matriz"() (matrices)
```

No realice ninguna modificación en el programa antes de verificar.

3.- LOAD   Se usa así: LOAD "nombre del programa"

Este comando, utilizado así, carga el programa BASIC que exista en la cinta con ese nombre, junto con sus variables y borra el programa y variables que hubiera en la memoria. Recuerde que ha de arrancar el cassette una vez escrito el comando. Si todo va bien, aparecerá la configuración de rayas de la que ya hemos hablado al tratar el comando SAVE, con la salvedad de que debe aparecer en la pantalla la palabra "Program:" seguida por el nombre del programa.

Si esto no sucede así, el programa no está cargando; en otras ocasiones aparecerá un mensaje "Tape Loading Error".

En ambos casos, rebobine la cinta, vuelva a introducir el comando e intente de nuevo la carga a diferente volumen (el volumen mas adecuado es, para un mando graduado del 1 al 10 el 8).

Si lo que se desea cargar es un bloque escrito en código máquina el comando será:

```
LOAD "nombre" CODE dirección,longitud
```

"dirección" especifica a partir de que posición de memoria se cargarán los datos. La longitud indica el número de bytes que se cargarán.

si se omite la longitud, se cargarán todos los bytes que se encuentren en el bloque. En cuanto a la dirección, si no se indica, los bytes se cargarán a partir de la dirección que se especificó cuando se grabó el bloque.

Antes de cargar el bloque, conviene hacer un CLEAR a la dirección "dirección" -1. Esto evitará que el bloque sea corrompido por algun programa en BASIC que carguemos o introduzcamos después.

La manera de cargar es la misma que la que se ha descrito para los programas en BASIC.

También puede usarse LOAD para cargar matrices previamente grabadas. El comando que se utiliza es:

```
LOAD "nombre" DATA "nombre de la matriz" ()
```

Atención: Tanto CODE como DATA son palabras clave del Spectrum.

2 - MERGE

Este comando se utiliza igual que LOAD, aunque solo sirve para programas BASIC. A diferencia de LOAD, no borra el programa que había en la memoria, sino que mezcla el procedente de la cinta con el existente. Tan solo se debe procurar que los números de linea de ambos no coincidan. Si así fuera, las lineas del programa procedente de la cinta desplazarían a las que existieran previamente con el mismo número.

Otra cualidad de MERGE es que si se utiliza para cargar un programa con autoejecución, ésta no se realiza.

Una observación: si se da la cadena vacía como nombre de programa con LOAD, VERIFY o MERGE, es decir:

LOAD "" (por ejemplo)

el comando actuará con el primer programa que encuentre en la cinta.

Los datos circulan del ordenador a la cassette y viceversa en serie, es decir, los bits del byte "desfilan" uno tras otro. La velocidad de transmisión se expresa en BAUDIOS, es decir, en bits por segundo. La velocidad de transferencia en el Spectrum es de 1.500 baudios, lo cual supone que un programa de 40 kbytes de longitud tardará unos 4 minutos en cargar.

Por otro lado, el cassette, a pesar de su economía, no es excesivamente fiable y tiende a dar errores de carga a menudo. Es por ello que existen otros dispositivos en el mercado que proporcionan mayor velocidad y fiabilidad.

La propia casa Sinclair ha comercializado un periférico llamado "microdrive" (se llama periférico a todo aquel aparato externo que puede conectarse a un ordenador) que permite la carga y grabación de datos a unos 15.000 baudios, a la vez que da mayor fiabilidad que el cassette. La informacién se graba en unas cintas sin fin de mucha mayor calidad que las de cassette, las cuales no necesitan ser posicionadas (rebobinadas o avanzadas) para cargar o grabar un programa; el propio ordenador se encarga de esa operación.

Los comandos para el uso del microdrive van incluidos ya en el teclado del Spectrum. Su utilización va claramente explicada en el manual que acompaña a estos equipos.

CAPITULO 21

EL FICHERO DE PANTALLA

En el capítulo dedicado a la memoria indicábamos que los primeros 6912 bytes de la memoria RAM albergaban el llamado "fichero de pantalla". Esta zona de la memoria almacena la imagen que se esta visualizando en la pantalla del televisor en cada momento.

Tambien decíamos que la imagen se almacenaba separada de los colores de la misma. Efectivamente, los últimos 768 bytes del fichero de pantalla contienen la información sobre los colores de la misma. Estos bytes se hallan codificados en función de los colores de papel y tinta de cada posición de carácter, de manera que con un solo número (almacenado en el byte correspondiente) podemos obtener toda la información sobre los colores de cada posición de carácter. A este número se le denomina "atributo" de esa posición, por lo que al conjunto de estos últimos 768 bytes se le suele denominar "fichero de atributos".

Los restantes bytes del fichero de pantalla almacenan la imagen propiamente dicha. Al contrario de lo que sucede en el fichero de atributos, los bytes correspondientes a la configuración de la imagen no se almacenan correlativamente. Efectivamente, en el fichero la pantalla es dividida en tres partes iguales: la superior, la intermedia y la inferior. Los primeros 256 bytes del fichero corresponden correlativamente a la primera linea de pixels de la primera linea de posiciones de carácter. Los 256 siguientes a la primera linea de pixels de la segunda linea de posiciones y lo mismo sucede con la tercera, cuarta y sucesivas lineas de posiciones de carácter del primer tercio de la pantalla. Luego se van rellenando las segundas lineas de pixels, despues las terceras lineas y así sucesivamente, hasta completar el primer tercio. La misma distribución se hace con los otros dos tercios de la pantalla.

Si todo esto puede parecerle un poco lioso (lo es) el siguiente programa le ayudara a comprenderlo. Escribalo y ejecútelo.

```
10 LOAD "" CODE 40000
20 FOR n=16384 TO 23296
30 POKE n,PEEK (n+23616): PAUSE 5
40 NEXT n
50 PAUSE 0
```

Coloque una cinta de juegos en el cassette. El segundo bloque grabado en la misma suele ser la "pantalla de presentación" que se ve en el televisor mientras carga el programa principal. Situe la cinta inmediatamente antes de dicho bloque y ejecute el programa.

Las lineas 20, 30 y 40 se encargan de transferir poco a poco el contenido de dicha pantalla (almacenada a partir de la dirección 40000) hasta el fichero de pantalla, con lo cual la imagen comenzará a aparecer en la pantalla del televisor en el mismo orden que se ha explicado antes.

La linea 50 detiene el programa para evitar que el mensaje "OK" estropee las dos ultimas lineas de la pantalla.

La trasferencia de datos mediante BASIC no es muy rápida, como habrá tenido ocasión de comprobar con este programa. El el último capítulo del libro, dedicado al código máquina, explicaremos una sencilla rutina que realiza esta operación en una fracción de segundo (!!!).

Debido a esta extraña distribución del fichero de pantalla, no resulta muy práctico gestionarlo por medio de PEEK y POKE, sobre todo teniendo en cuenta que existen comandos en el BASIC del Spectrum que se encargan de ello facilmente (PRINT AT, PLOT, etc.).

La función SCREEN$ detecta cualquier carácter situado en una posición determinada de la pantalla. Su argumento es el número de linea y el de columna que nos interesa y su resultado es una cadena que contiene el carácter detectado por la función.

```
10 PRINT AT 11,5;"Spectrum"
20 PRINT SCREEN$ (11,7)
```

Existe un inconveniente: SCREEN$ no detecta los G.D.U.

Si la gestión del fichero de imagen por medio de PEEK y POKE no es práctica no ocurre lo mismo con el fichero de atributos. Existe una función , denominada ATTR, cuyo argumento es la posición de caracter que nos interesa y su resultado el número correspondiente del fichero de atributos.

El atributo se calcula de la siguiente forma:

atributo=numero color tinta+8*numero color papel+128 (si existe parpadeo)+64 (si existe brillo especial).

La función ATTR nos permite calcular el atributo de una posición de caracter dada, pero por desgracia no existe en el Spectrum un comando que haga lo contrario (cambiar el atributo de una posición de carácter). Sin embargo ello no implica que no podamos simularla.

El siguiente programa imprime un texto en el lugar que queramos y en un color determinado, y luego cambia los colores de lo impreso sin afectar para nada lo que está escrito.

```
10 DEF FN a(x,y)=22528+(x*32)+y
20 DEF FN b(i,p,b,f)=i+(p*8)+128*(f=1)+64*(b=1)
30 INPUT "Posicion x ";x,"Posicion y ";y
40 INPUT "Texto ";a$
50 INPUT "Papel ";p,"Tinta ";i
60 INPUT "Brillo (0-1) ";b,"Flash (0-1) ";f
70 PRINT AT x,y;INK 7;PAPER 1;a$
80 PAUSE 200
90 FOR n=1 TO LEN a$
100 POKE FN a(x,y),FN b(i,p,b,f)
110 LET y=y+1
120 NEXT n
```

El texto, que originalmente estaba escrito en papel azul y tinta blanca, cambia a los valores que ha introducido, sin que se altere para nada lo que había escrito.

CAPITULO 22

USO DE LA IMPRESORA

El Spectrum no solo puede mandar datos a la pantalla o al grabador de cassette; tambien puede escribirlos sobre papel con la ayuda de una impresora. Existen diversos modelos de impresora que pueden conectarse, bien directamente, bien a través de una "interface".

La interface es un dispositivo que permite al ordenador conectarse con otros dispositivos ajenos al mismo. Las interfaces de impresora pueden ser de dos tipos: serie y paralelo. En las primeras, como en el cassette, los bits integrantes del byte son enviados uno detras del otro. El tipo de interface serie mas extendido es el RS-232.

La interface paralelo no realiza el envio bit a bit sino byte a byte; es decir, los ocho bits del byte son procesados a la vez. El tipo mas extendido es el Centronics. En realidad, las impresoras suelen estar adaptadas mas al sistema paralelo, mientras que las interfaces serie se emplean preferiblemente en otros periféricos.

Las interfaces suelen llevar un programa en código máquina incorporado que se encarga de establecer un "protocolo" de comunicación con la impresora. Este programa puede estar grabado en ROM dentro de la misma interface o bién registrado en una cinta, cuyo contenido hay que cargar en la RAM del ordenador después de conectada la interface.

Los comandos que se utilizan para imprimir con el ZX Spectrum son LPRINT, LLIST y COPY. Los dos primeros sirven como sustitutos de PRINT y LIST para que estos comandos actuen en la impresora.

El Spectrum transmite los caracteres previamente almacenados en el "buffer" de impresora a la misma a traves del canal "p". Este canal emplea una linea (o "corriente") para la transmisión de los datos (la 3) por lo que LPRINT y LLIST pueden ser sustituidos por PRINT # 3 o LIST # 3.

COPY realiza un volcado del fichero de pantalla (excepto los atributos), por lo que se obtendrá en papel una imagen de la pantalla de televisión.

Solo puede usarse TAB como comando complementario de LPRINT, ya qu AT no es interpretado por la impresora. Se puede especificar en LLIST el numero de linea desde donde hay que empezar el listado (igual que en LIST).

Algunas interfaces premiten realizar un COPY a doble tamaño.

CAPITULO 23

LOS "PORTS" DE ENTRADA-SALIDA

Los "ports" de E/S son las lineas de comunicación que utiliza el microprocesador Z-80 del Spectrum para comunicarse con los diversos dispositivos periféricos. Entre estos dispositivos están incluidos aparatos tales como el teclado, la pantalla y el altavoz, que normalmente creemos parte integrante del propio ordenador.

El procesador comunica con el mundo exterior por medio de los "buses". Hay tres buses en el sistema: el bus de direcciones, que dispone de 16 bits, el de datos, de ocho bits y el de control del propio sistema, de 6 bits. Los ports se direccionan con la mitad inferior del bus de direcciones, es decir, con los últimos 8 bits; por lo tanto, podremos definir 256 ports diferentes.

Cada uno de los bits del port controla un periférico diferente. Para indicar que periférico estamos operando, el bit correspondiente al mismo deberá estar a 0 y todos los demás a 1.

Así, el port 254 (binario 11111110) maneja el teclado y la entrada EAR cuando se usa como lectura y el altavoz, la salida MIC y el borde de la pantalla cuando se usa como escritura (o salida).

Los comandos que tratan con los ports son la sentencia OUT y la función IN. La primera escribe un dato en el port correspondiente y la segunda lee un dato de ese port.

Teclee lo siguiente:

```
10 FOR n=1 TO 25
20 OUT 254, n
30 NEXT n
40 GO TO 10
```

para detener el programa pulse BREAK.

Cuando se lee el teclado por medio de la funcion IN se emplean los 16 bits del bus de direcciones, ya que los numeros que son argumentos de IN son mayores de 256.

Eso es porque, aunque solo pueden direccionarse 256 ports, puede utilizarse la mitad superior del bus de direcciones para escribir números complementarios que indiquen a la ULA lo que tiene que hacer (dispone de referencias a la ULA en la cinta de demostración).

La conbinación de ambas mitades del bus de direcciones constituyen un numero binario que, traducido al decimal constituye el argumento para IN. Por ejemplo si escribimos PRINT IN 65022, el número significa que se escribe 254 en la mitad inferior del bus de direcciones (el port que se encarga de la lectura del teclado) y 253 en la superior (indicativo de la semifila a leer, en este caso de la "a" a la "g".

CAPITULO 24

LAS VARIABLES DEL SISTEMA

En la memoria RAM, inmediatamente despues del "buffer" de impresora, se encuentra la zona reservada a las variables del sistema, cuyo valor es determinado por el propio ordenador según sus necesidades. Esto no quiere decir que no podamos variar nosotros mismos esos valores para adaptarlos a las tareas que queramos realizar.

Veamos ahora una descripción de estas variables en el orden en que se encuentran en la memoria.

Nombre: KSTATE　　Dirección: 23552　　Longitud: 8 bytes

Cada uno de los bytes de esta variable tiene una función especial, pero en esencia lo que hacen es leer la información suministrada por el teclado. Cada byte almacena valores correspondientes a la tecla que acaba de pulsarse y el tiempo que ha de transcurrir antes de que se autorrepita.

Nombre: LASTK　　Dirección: 23560　　Longitud: 1 byte

Almacena el valor de la ultima tecla pulsada. Es la variable que utiliza la función INKEY$.

Nombre: REPDEL　　Dirección: 23561　　Longitud: 1 byte

Controla el tiempo que se ha de tener pulsada una tecla antes de que comience la autorrepetición. Su valor inicial es 35 pero puede alterarse para aumentarlo o disminuirlo.

Nombre: REPPER　　Dirección: 23562　　Longitud: 1 byte

Almacena un valor para controlar el tiempo entre autorrepeticiones. Inicialmente vale 5, pero también puede alterarse.

Nombre: DEFADD　　Dirección: 23563　　Longitud: 2 bytes

Almacena la dirección donde se encuentra el argumento de una función definida por el usuario. Si no hay ninguna su valor es 0.

Nombre: K DATA Dirección: 23565 Longitud: 1 byte

Almacena el segundo byte del control de color que se ha introducido por el teclado.

Nombre: TVDATA Dirección: 23566 Longitud: 2 bytes

Almacena el primer byte del control del color y controles para AT y TAB.

Nombre: STRMS Dirección: 23568 Longitud: 38 bytes

Almacena las direcciones de los canales de comunicación.

Nombre: CHARS Dirección: 23606 Longitud: 2 bytes

Contiene una dirección que es igual a la dirección donde comienza el juego de carcteres (desde "espacio" hasta "Copyright") menos 256. Los valores de los 2 bytes son 0 y 60, que es la dirección de la ROM donde esta almacenado el juego de caracteres standard, pero puede alterarse para que el ordenador se dirija a otra dirección de la RAM donde hayamos definido nuestro propio juego de caracteres, o incluso sustituirlo por los G.D.U.

Nombre: RASP Dirección: 23608 Longitud: 1 byte

Indica la duración del zumbido que suena cuando se ha llenado la memoria. Puede alterarse y su valor normal es 64.

Nombre: PIP Dirección: 23609 Longitud: 1 byte

Indica la duración del pitido que se produce cuando se pulsa una tecla. Inicialmente vale 0 por lo que suena como un chasquido. El valor recomendado para un pitido audible es 50; si se le da un valor superior de puede enlentecer demasiado la entrada de datos.

Nombre: ERR NR Dirección: 23610 Longitud: 1 byte

Almacena el codigo del mensaje de error correspondiente. Si no hay ningún error vale 255 y si lo hay almacena el código de error menos 1.

Nombre: FLAGS Dirección: 23611 Longitud: 1 byte

Cada uno de los bits del byte es un indicador (banderín) que indica una cosa u otra segun este a 0 o a 1.

Nombre: TVFLAGS Dirección: 23612 Longitud: 1 byte

Igual que el anterior pero asociado con la pantalla de la televisión.

Nombre: ERR SP Dirección: 23613 Longitud: 2 bytes

Almacena una direccion de memoria que es donde tiene que dirigirse la ejecución en caso de error. Si introducimos un 0 en esta variable, el ordenador se borrará al producirse cualquier error.

Nombre: LIST SP Dirección: 23615 Longitud: 2 bytes

Direccion a la que hay que volver despues de un listado automático.

Nombre: MODE Dirección: 23617 Longitud: 1 byte

Indica la letra o símbolo que tiene que aparecer en el cursor. Puede alterarse para que aparezca cualquier cosa, pero eso no afectará al auténtico modo en que se encuentre el ordenador.

Nombre: NEWPPC Dirección: 23618 Longitud: 2 bytes

Almacena el numero de linea al que hay que saltar. Podemos simular el comando GO TO alamacenando en esta variable en número de linea al que queremos saltar.

Nombre: NSPPC Dirección: 23620 Longitud: 1 byte

Almacena el número de sentencia al que hay que saltar. Se emplea conjuntamente con la anterior.

Nombre: PPC Dirección: 23621 Longitud: 2 bytes

Almacena el número de la linea en curso.

Nombre: SUBPPC Dirección: 23623 Longitud: 1 byte

Almacena número de sentencia que se esta ejecutando dentro de la linea en curso.

Nombre: BORDCR Dirección: 23624 Longitud: 1 byte

Almacena los atributos de la parte inferior de la pantalla y el color del BORDER.

Nombre: E PPC Dirección: 23625 Longitud: 2 bytes

Almacena el número de la linea en curso (donde está situado el cursor del programa).

Nombre: VARS Dirección: 23627 Longitud: 2 bytes

Almacena la dirección donde comienza la zona de variables.

Nombre: DEST Dirección: 23629 Longitud: 2 bytes

Almacena la dirección donde se almacena una variable al ser asignada.

Nombre: CHANS Dirección: 23631 Longitud: 2 bytes

Almacena la dirección de comienzo del canal de información.

Nombre: CURCHL Dirección: 23633 Longitud: 2 bytes

Almacena la dirección donde empieza la información de entrada/salida.

Nombre: PROG Dirección: 23635 Longitud: 2 bytes

Contiene la dirección de comienzo del programa BASIC.

Nombre: NXTLIN Dirección: 23637 Longitud: 2 bytes

Guarda la dirección de comienzo de la siguiente linea.

Nombre: DATADD Dirección: 23639 Longitud: 2 bytes

Contiene la direccion de la terminacion de los datos (los que se leen mediante READ).

Nombre: ELINE Dirección: 23641 Longitud: 2 bytes

Indica la direccion donde acaban las variables.

Nombre: STKEND   Dirección: 23653   Longitud: 2 bytes

Contiene la dirección donde empieza la memoria libre.

Nombre: FLAGS2   Dirección: 23658   Longitud: 1 byte

Contiene mas indicadores similares a los de FLAGS. El mas importante es el bit 3, que indica si está activado o no CAPS LOCK. Para utilizar esta posibilidad, teclee 23658,8 para activar el modo C y 23658,0 para desactivarlo.

Nombre: DF SZ   Dirección: 23659   Longitud: 1 byte

Indica el número de lineas que hay en la parte inferior de la pantalla.

Nombre: S TOP   Dirección: 23660   Longitud: 2 bytes

Almacena el número de la linea superior en los listados automáticos.

Nombre: OLDPPC   Dirección: 23662   Longitud: 2 bytes

Número de linea a la que salta CONTINUE

Nombre: OSPCC   Dirección: 23664   Longitud: 1 byte

Contiene el número de la sentencia dentro de la correspondiente linea a la que salta CONTINUE.

Nombre: SEED   Dirección: 23670   Longitud: 2 bytes

Contiene el número de orden en la serie de números pseudoaleatorios desde el que empieza a actuar RND (se fija mediante RANDOMIZE).

Nombre: FRAMES   Dirección: 23672   Longitud: 3 bytes

Cuenta el numero de veces que cambia la pantalla de televisión (aproximadamente 50 veces por segundo). Para leer esta variable no se emplea la formula usual (indicada en el capítulo dedicado a la memoria), ya que el orden de significación de los bytes no es el mismo. Usaremos:

```
PRINT 65636*PEEK 23674+256*PEEK 23673+PEEK 23672
```

Esto puede servirnos como reloj, pero hay que tener en cuenta que el comando BEEP y las operaciones con el cassette "paran" dicho reloj.

Nombre: UDG Dirección: 23675 Longitud: 2 bytes

Dirección donde comienzan los G.D.U.

Nombre: COORDS Dirección: 23677 Longitud: 2 bytes

El primer byte contiene la coordenada x y el segundo la coordenada y de la posición PLOT.

Nombre: P POSN Dirección: 23679 Longitud: 1 byte

Número de columnas a imprimir en la impresora.

Nombre: PRCC Dirección: 23680 Longitud: 2 bytes

Menor y mayor byte, respectivamente, de la posición LPRINT en el "buffer" de la impresora.

Nombre: DF CC Dirección: 23684 Longitud: 2 bytes

Contiene la dirección de la posición de PRINT en el fichero de pantalla.

Nombre: SPOSN Dirección: 23688 Longitud: 2 bytes

Indica en cada momento donde está situada la posición PRINT (columna y linea respectivamente) en la pantalla.

Nombre: SCR CT Dirección: 23692 Longitud: 1 byte

Indica al ordenador los desplazamientos hacia arriba del contenido de la pantalla que tiene que hacer antes de preguntar "scroll?". Inicialmente vale 1, pero pueden usarse numeros mayores para evitar el mensaje.

Nombre: ATTR P Dirección: 23693 Longitud: 1 byte

Comtiene los atributos generales de la pantalla (no los que se especifican en sentencias PRINT).

Nombre: MASK P Dirección: 23694 Longitud: 1 byte

Cada bit de esta variable se corresponde con otro en ATTR P. Cuando se utiliza el valor 8 en comandos de color, se pone a 0 el bit correspondiente de esta variable.

Nombre: ATTR T Dirección: 23695 Longitud: 1 byte.

Controla los atributos que se especifican para cada sentencia PRINT en particular (al contrario de ATTR P).

Nombre: MASK T Dirección: 23696 Longitud: 1 byte

Igual que MASK P pero con respecto a ATTR T.

Nombre: P FLAGS Dirección: 23697 Longitud: 1 byte

Especifica el uso de los comandos PAPER 9, INK 9, INVERSE y OVER.

Nombre: RAMTOP Dirección: 23730 Longitud: 2 bytes

Indica la dirección del ultimo byte del area del BASIC. Se puede especificar mediante CLEAR. NEW solo opera hasta esta dirección.

Nombre: RAMTP Dirección: 23732 Longitud: 2 bytes

Dirección donde acaba la RAM del Spectrum.

Recuerde que solo puede cambiar sin peligro aquellas variables que le hemos indicado. Si juega con las otras corre el riesgo de que el sistema quede sin control o bloqueado. Por ejemplo, si emplea POKE 23659,0 para tener 24 lineas en la pantalla, cualquier mensaje de error, sentencia INPUT o cualquier otro comando que afecte a la parte inferior de la pantalla hara que el ordenador se bloquee sin remedio.

CAPITULO 25

INTRODUCCION A LA PROGRAMACION EN LENGUAJE MAQUINA

Al principio de este libro indicábamos que el BASIC y los otros lenguajes de programación se habían inventado dada la dificultad que existía en hablar al ordenador en su propio lenguaje. Como sabemos, el procesador Z 80 de que está dotado el Spectrum (todos los ordenadores poseen un procesador, que no tiene por que ser un Z 80) solo entiende números. En realidad, la información la obtiene de sus memorias RAM y ROM, en cuyas diversas posiciones hay almacenados números entre 0 y 255, como ya sabemos.

Si en lugar de utilizar el BASIC introducimos los correspondientes números en las direcciones adecuadas mediante POKE, estaremos utilizando el mismo lenguaje que emplea la máquina; estaremos, pués, programando en "lenguaje máquina". Esto tiene la ventaja de que el ordenador se ahorra el tiempo de la traducción que realiza mediante el BASIC, por lo que la velocidad de ejecución de los programas en código máquina suelen ser unas 300 veces más rápida que la de los programas en BASIC.

Existe un lenguaje intermedio entre los de alto nivel y el código máquina, llamado "ensamblador". Cada procesador tiene su propio ensamblador (es decir, el ensamblador que utilicemos en el Spectrum es el mismo que usaríamos en cualquier otro ordenador que disponga del mismo procesador). En el ensamblador se usan unos comandos abreviados que llamamos "mnemónicos" que actúan sobre el mismo procesador.

Para ilustrar todo esto, vamos a utilizar una pequeña rutina en lenguaje máquina, que nos servirá para trasplantar una pantalla almacenada en la memoria a la zona del fichero de representación visual en una fracción de segundo. En el capítulo dedicado al fichero de pantalla veíamos como se hacía ésto mediante el BASIC.

La rutina tiene 12 bytes de longitud y la vamos a almacenar a partir de la dirección de memoria 60000.

```
10 CLEAR 39999
20 LOAD "" CODE 40000
30 DATA 17,0,64,33,64,156,1,0,27,237,176,201
40 FOR n=60000 TO 60011
50 READ a: POKE n,a: NEXT n
60 PAUSE 0: RANDOMIZE USR 60000
```

Colocamos el RAMTOP en 39999 para que la pantalla que vamos a cargar en esa dirección no resulte afectada por el BASIC. A continuación cargamos la pantalla e introducimos el programa en código máquina con la ayuda del bucle FOR - NEXT y las sentencias READ - DATA.

El PAUSE 0 evita que el programa en lenguaje máquina se ejecute inmediatamente. En cuanto al comando RANDOMIZE USR, equivale al RUN del BASIC; hace que se ejecute el programa en código máquina existente a partir de la dirección especificada.

El listado del programa en lenguaje ensamblador sería como sigue:

```
LD DE, 16384
LD HL, 40000
LD BC, 6912
LDIR
RET
```

Esto se traduce como sigue:

Carga el par de registros DE con el número 16384
Carga el par de registros HL con el número 40000
Carga el par de registros BC con el número 6912
Carga, incrementa y repite
Vuelve al BASIC

Los pares de registros DE, HL y BC existen en el procesador. Como cada registro tiene 8 bits, un par de registros tendrá 16 bits, luego podrá almacenar un número comprendido entre 0 y 65535. Los números 17, 33 y 1 significan "LD DE", "LD HL" y "LD BC" respectivamente. En cuanto a los pares de numeros 0-64, 64-156 y 0-27, si se les aplica la fórmula:

primer número+256*segundo número

veremos que resulta, respectivamente, 16384, 40000 y 61912, que son, por este orden, la nueva dirección a donde se trasplanta el bloque, la dirección de comienzo actual y la longitud del mismo.

Los números 237 y 176 indican la instrucción propiamente dicha (la que realiza el trasplante) y 201 hace que se vuelva al BASIC cuando el programa ha terminado de ejecutarse. Esta instrucción es absolutamente necesario colocarla al final del programa si no queremos que el ordenador se nos bloquee.

Ejecute el programa y vea lo que sucede.

En el mercado existen programas ensambladores que permiten la programacion en código máquina por medio de comandos. Si no quiere utilizar ensamblador, hágase con un libro que explique el ensamblador del Z 80 y traduzca Vd. mismo el programa a lenguaje máquina, aunque seguramente encontrará muy tedioso este método.

La manera de grabar programas en código máquina se explicó en el capítulo correspondiente.

APENDICE

EL JUEGO DE CARACTERES

A continuación damos la relación del juego de caracteres del 32 al 255 (el resto no admite impresión) con sus correspondientes códigos.

CARACTER	CODIGO
espacio	32
!	33
"	34
#	35
$	36
%	37
&	38
'	39
(	40
)	41
*	42
+	43
,	44
-	45
.	46
/	47
0	48
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57
:	58
;	59
<	60
=	61
>	62
?	63
§	64
A	65

B	66
C	67
D	68
E	69
F	70
G	71
H	72
I	73
J	74
K	75
L	76
M	77
N	78
O	79
P	80
Q	81
R	82
S	83
T	84
U	85
V	86
W	87
X	88
Y	89
Z	90
[	91
/	92
]	93
↑	94
_	95
£	96
a	97
b	98
c	99
d	100
e	101
f	102
g	103
h	104
i	105
j	106
k	107
l	108
m	109

n	110
o	111
p	112
q	113
r	114
s	115
t	116
u	117
v	118
w	119
x	120
y	121
z	122
{	123
|	124
}	125
~	126
©	127
gráfico 8	128
gráfico 1	129
gráfico 2	130
gráfico 3	131
gráfico 4	132
gráfico 5	133
gráfico 6	134
gráfico 7	135
inverso 7	136
inverso 6	137
inverso 5	138
inverso 4	139
inverso 3	140
inverso 2	141
inverso 1	142
inverso 0	143
gráfico A	144
gráfico B	145
gráfico C	146
gráfico D	147
gráfico E	148
gráfico F	149
gráfico G	150
gráfico H	151
gráfico I	152
gráfico J	153

gráfico K	154
gráfico L	155
gráfico M	156
gráfico N	157
gráfico O	158
gráfico P	159
gráfico Q	160
gráfico R	161
gráfico S	162
gráfico T	163
gráfico U	164
RND	165
INKEY$	166
PI	167
FN	168
POINT	169
SCREEN$	170
ATTR	171
AT	172
TAB	173
VAL$	174
CODE	175
VAL	176
LEN	177
SIN	178
COS	179
TAN	180
ASN	181
ACS	182
ATN	183
LN	184
EXP	185
INT	186
SQR	187
SGN	188
ABS	189
PEEK	190
IN	191
USR	192
STR$	193
CHR$	194
NOT	195
BIN	196
OR	197

AND	198
<=	199
>=	200
<>	201
LINE	202
THEN	203
TO	204
STEP	205
DEF FN	206
CAT	207
FORMAT	208
MOVE	209
ERASE	210
OPEN #	211
CLOSE #	212
MERGE	213
VERIFY	214
BEEP	215
CIRCLE	216
INK	217
PAPER	218
FLASH	219
BRIGHT	220
INVERSE	221
OVER	222
OUT	223
LPRINT	224
LLIST	225
STOP	226
READ	227
DATA	228
RESTORE	229
NEW	230
BORDER	231
CONTINUE	232
DIM	233
REM	234
FOR	235
GO TO	236
GOSUB	237
INPUT	238
LOAD	239
LIST	240
LET	241

PAUSE	242
NEXT	243
POKE	244
PRINT	245
PLOT	246
RUN	247
SAVE	248
RANDOMIZE	249
IF	250
CLS	251
DRAW	252
CLEAR	253
RETURN	254
COPY	255

INDICE


MICRO
MW
WORLD